Escola d'Enginyeria de Telecomunicació i Aeroespacial de Castelldefels
UNIVERSITAT POLITÈCNICA DE CATALUNYA

TREBALL FINAL DE GRAU

TÍTOL DEL TFG: Millora d'una eina de gamificació

TITULACIÓ: Grau en Enginyeria Telemàtica

AUTOR: Alejandro Martín Cruz

DIRECTOR: Miguel Valero Garcia
CODIRECTOR: Roque Messeguer Pallares

DATA: 05 de febrer de 2019

Dedicat a la meva filla Maria, que m'ha proporcionat la motivació necessària per a realitzar i acabar aquest projecte, i a la meva dona Alba, que m'ha recolzat durant tot el temps que ha durat.

Títol: Millora d'una eina de gamificació

Autor: Alejandro Martín Cruz

Director: Miguel Valero Garcia
Codirector: Roque Messeguer Pallares

Data: 05 de febrer de 2019

Resum

La gamificació es la introducció de mecàniques dels jocs en àmbits que no estan considerats lúdics, creant una motivació per a realitzar les coses establint uns objectius i atorgant premis per aconseguir-los. Classpip es una eina que aplica la gamificació a l'entorn educatiu.

Classpip consta de tres parts, una aplicació web, una aplicació mòbil per a iOS/Android i una aplicació que garanteix la comunicació de les dues plataformes i la persistència de les dades.

Aquest treball s'ha fonamentat en la millora i ampliació del projecte de Classpip, afegint algunes noves funcionalitats com la de rangs i nivells o la de Rewards, i realitzant diversos canvis en el codi que afavoreixen tant la usabilitat de l'aplicació com el seu posterior manteniment. Per tal d'afavorir l'adaptació al projecte de futurs estudiants, també s'ha afegit al projecte una nova aplicació web que a més de recol·lectar tot el material didàctic sobre el projecte també serveix com a projecció externa per a aquest. Les darreres adicions al treball han sigut un manual per a facilitar el treball en equip mitjançant Git, i un manual per a desenvolupar canvis i millores a la nova pàgina web.

L'objectiu final ha estat tenir una versió funcional que es pugui arribar a utilitzar a escoles.

Title: Improvement of a gamification tool

Author: Alejandro Martín Cruz
Director: Miguel Valero Garcia
Codirector: Roque Messeguer Pallares

Date: February 5th 2019

Overview

Gamification is the introduction of game mechanics in environments that are not considered playful, creating a motivation to do activities by setting goals and awarding prizes to achieve them. Classpip is a tool that applies gamification to the educational environment.

Classpip consists of three parts, a web application, a mobile application for iOS/Android and an application that guarantees the communication of both platforms and the persistence of the data.

This work has been based on the improvement and expansion of the Classpip project, adding some new features such as ranges and levels or the Rewards one, and making several changes in the code that improves the usability of the application and its subsequent maintenance. In order to encourage adaptation to the future students in the project, a new web application has also been developed. This web collects all the teaching material about the project, and it also serves as an external projection for the project. The latest additions to work have been a manual to facilitate teamwork through Git, and a manual to develop changes and improvements to the new website.

The final objective was to have a functional version that can be used at schools.

ÍNDEX

INTRODUCCIÓ

A dia d'avui, un dels problemes més greus en els entorns educatius es la manca de motivació a l'hora d'estudiar. Un estudiant sense cap motivació, no tindrà prou interès per aprendre les matèries.

Hi ha moltes formes de despertar l'interès dels estudiants vers l'entorn educatiu, però potser una de les més innovadores, i potser més senzilles d'aplicar, es la introducció de la **Gamificació**.

La gamificació consisteix en la aplicació a classe, d'algunes de les mecàniques dels jocs. Com per exemple, les que en els darrers anys s'han aplicat als videojocs, com poden ser els punts, trofeus, col·leccions, etc.

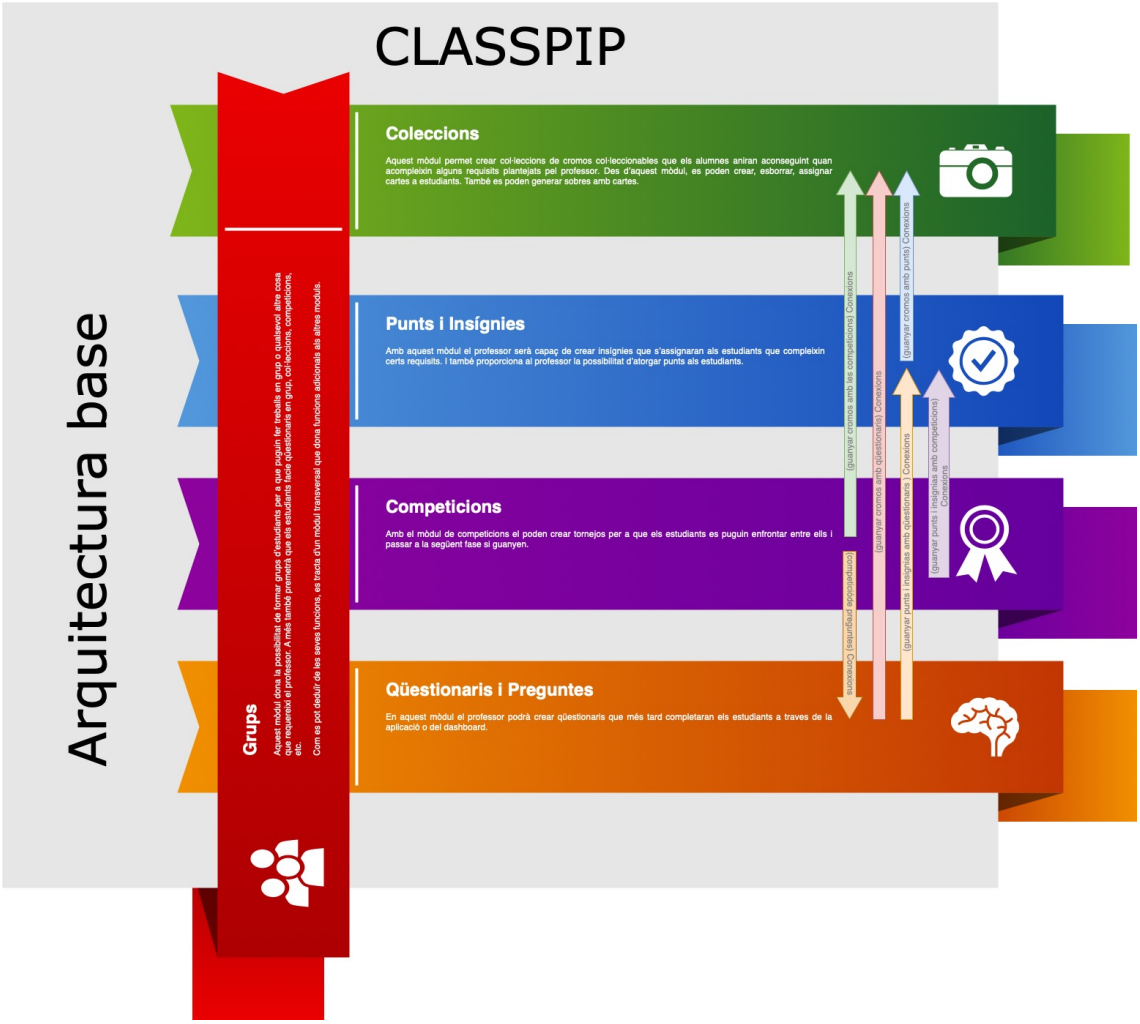
Amb l'arribada de les noves tecnologies de la informació a les aules, es possible aplicar la gamificació de forma fàcil i entenedora, tant per a estudiants com per a professors. De fet, ja hi ha aplicacions que apliquen tècniques de gamificació. N'hi ha aplicacions que permeten atorgar punts als estudiants per realitzar algunes accions, n'hi ha aplicacions que recompensen els alumnes amb insígnies, n'hi ha aplicacions que permeten fer qüestionaris amb les respostes del qual es poden fer rankings dels millors, n'hi ha aplicacions que permeten fer col·leccions de cromos que es reparteixen als estudiants per aconseguir certes fites. N'hi ha moltes aplicacions, cada una centrada en un àmbit concret de la gamificació. I per difícil que sigui de creure, encara no hi ha cap aplicació de gamificació que reuneixi totes aquestes eines alhora en una sola aplicació.

Amb aquest objectiu, s'ha creat una eina per augmentar la motivació dels estudiants de forma que tampoc suposi un greu problema per als professors, que a més reuneixi eines de gamificació que fins ara han estat disperses en diverses aplicacions. Aquesta eina, anomenada **Classpip**, consisteix en una pagina web i una aplicació mòbil per a Android i iOS, que seran fàcilment utilitzables a classe amb portàtils, mòbils i tabletes.

Tot l'entorn de Classpip ha estat desenvolupat per estudiants de la EETAC en projectes de final de màster o de final de grau. Cadascun d'aquests projectes tractaven temes diferents com la base de l'arquitectura, o diferents funcionalitats com les de Col·leccions, Punts i insígnies, Preguntes i Grups. L'actual arquitectura del projecte la podem veure a la il·lustració 1.

Ara mateix Classpip es la suma de tots aquests projectes, creant un macroprojecte on s'han integrat tots aquests projectes individuals, fins i tot creant connexions que fan que hi hagi algunes interaccions entre les diferents funcionalitats.

L'objectiu principal d'aquest projecte es solucionar un problema que s'ha generat amb tots els projectes anteriors, i es que cada projecte te un repositori separat dels anteriors, la documentació esta dispersa en diversos documents, i es una mica difícil introduir-se a un projecte tan gran on no hi ha una estructura ordenada de la documentació que faciliti trobar informació de forma clara i entenedora.



Il·lustració 1: components del projecte

Així, com a part del projecte, es crearà una pàgina web amb Angular on tota la documentació estigui ordenada, visible, fàcil de trobar i de comprendre. I tot a un mateix lloc.

Un altre dels objectius del projecte, es crear una metodologia de treball amb git per a fer que tots els estudiants treballin en un únic repositori, i que faciliti la integració entre diversos projectes.

Per acabar, també s'han afegit objectius de millora en el codi en les aplicacions de Classpip. Aquests canvis en el codi, es resumeixen en: millores d'usabilitat en el dashboard i en mobile, diverses millores de codi i ampliar el mòdul de punts i insígnies amb la possibilitat d'assignar nivells segons els punts aconseguits.

L'estructura del projecte es resumeix en:

- Una introducció a la gamificació, on s'explica de que tracta i es mostren alguns exemples reals.
- Una iniciació al projecte, on s'explica l'objectiu de l'aplicació, l'estructura que s'ha utilitzat, i les tecnologies emprades per al seu desenvolupament.
- Una secció on s'expliquen noves funcionalitats desenvolupades. Concretament una funcionalitat de rangs i nivells i una altre funcionalitat que s'ha anomenat Rewards i que serveix per a mostrar informació dels avenços dels estudiants més fàcilment.
- Una secció on s'explica el desenvolupament d'una pàgina web, que passarà a formar part del projecte, i que reunirà el material d'onboarding on es donen algunes explicacions sobre l'entorn de desenvolupament, com iniciar les aplicacions del projecte, i errors comuns que s'han trobat durant el projecte. A més d'oferir un cert grau de projecció externa amb una landing que ofereix alguna informació sobre el projecte de Classpip.
- Una secció on s'expliquen els canvis i millores de codi realitzats al la part del dashboard del projecte i que avantatges tenen aquest canvis sobre el que hi havia abans.
- I per acabar en els annexos s'adjuntaran tutorials amb una metodologia de treball de git, i amb un manual de desenvolupament pel projecte de la web de documentació i projecció externa.

Un cop introduït el concepte bàsic del projecte, caldria explicar millor què es la gamificació, com es pot aplicar en entorns educatius i quins avantatges aporta. Aquestes aclaracions tot just es troben al pròxim capítol.

CAPÍTOL 1. LA GAMIFICACIÓ

Tenint ja un concepte bàsic sobre el projecte, cal remarcar que Gamificació es un terme molt desconegut per a la majoria de la gent, per tant cal una explicació d'aquest terme per a facilitar la comprensió general del projecte.

1.1 Què és la Gamificació ?

Gamificació es una paraula que prové de l'anglès: *'gamification'*, o el que seria una traducció més acurada: 'convertir en joc'. Al nostre idioma també es conegut per *'ludificació'*. I es l'aplicació de les mecàniques dels jocs en entorns educatius o professionals amb la finalitat de motivar els usuaris a participar en les activitats.

En els últims anys hem vist proliferar en el món dels videojocs nous mètodes per a atreure jugadors de forma habitual. D'aquesta manera, s'han introduït perfils de jugadors, on entre d'altres coses, aquests poden obtenir trofeus quan aconsegueixen realitzar certes fites als videojocs, i aquests trofeus aporten uns punts al perfil del jugador. Segons el jugador va guanyant punts, va aconseguint un nivell. Finalment el jugador pot compartir els trofeus que ha aconseguit, o el seu nivell, a les xarxes socials per poder presumir d'haver aconseguit gestes molt difícils en algun videojoc que pocs jugadors més han aconseguit.

Amb aquests mecanismes de gamificació, s'ha aconseguit que molta gent incrementi els seus esforços considerablement per a aconseguir nivell i prestigi i poder compartir-ho a les xarxes socials, arribant a estar fins a altes hores de la nit per a poder aconseguir insígnies, punts o qualsevol altre cosa que augmenti el seu prestigi enfront els seus companys.

La implantació de la gamificació en l'entorn educatiu crea metes i objectius que pretenen motivar als estudiants, que hauran de superar diferents reptes que els diferenciaran dels seus companys, creant una rivalitat constructiva entre ells que els impulsi a millorar el seu rendiment acadèmic.

Molts docents destaquen el caràcter positiu en els alumnes de la utilització d'aquests mètodes, que utilitzen per a potenciar i treballar aspectes com la motivació, l'esforç, la fidelització i la cooperació.

Alguns dels punts principals que es podrien considerar imprescindibles per a la aplicació de la gamificació en un entorn educatiu, i s'aniran remarcant durant la resta del projecte, son:

Recompenses

Igual que s'ha fet als videojocs, s'ha de crear un sistema de recompenses. Aquest, ha de proposar uns **objectius** que es puguin anar assolint de forma progressiva, i que a mesura que es vagin aconseguint proporcionin **premis**, es a dir, una gratificació en forma de punts, nivells, cromos, insígnies, etc que es puguin exhibir davant dels altres estudiants.

Classificació

Les competicions son unes de les principals motivacions dels jocs, per aquesta raó es molt important que els alumnes puguin comparar els seus avenços amb els dels altres estudiants mitjançant algun tipus de ranking. Aquest ranking pot incloure in sistema de rangs, que són imatges identificatives i títols que els alumnes aconsegueixen amb els seus avenços.

Sistema de nivells de dificultat

A la majoria dels jocs el nivell de dificultat augmenta de forma progressiva segons es va avençant. I a les activitats educatives, aquest comportament també s'ha de replicar. Per aquesta raó els **objectius** a aconseguir cada cop han de ser més complicats, o fins i tot poden requerir d'haver aconseguit prèviament altres objectius més senzills abans de que l'estudiant pugui aconseguir els més difícils.

Percentatges

Una de les coses que poden motivar molt en els jocs es veure el percentatge que resta per a finalitzar una activitat completament. Les barres de percentatges sempre inviten a prosseguir amb els esforços fins a enllestir-les del tot.

Objectius i premis

S'han de fixar objectius clars als estudiants, amb els quals podran aconseguir les **recompenses** disponibles com a premis. Atorgar les recompenses quan s'assoleixen els objectius es un clar motivador per als estudiants.

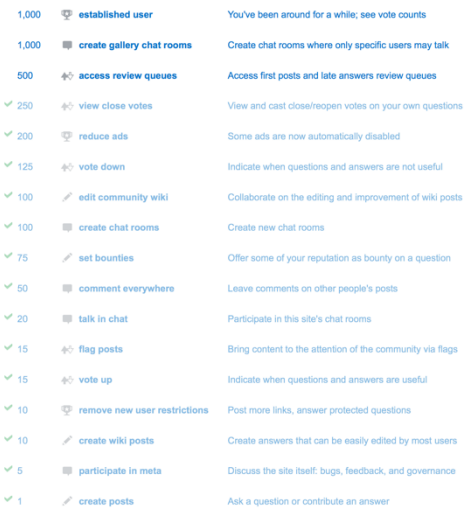
1.2 Exemple real de la Gamificació

Un exemple perfecte de l'aplicació de la gamificació, es pot veure a la pagina d'stackoverflow. Aquesta pàgina es la major comunitat de desenvolupadors informàtics del mon, i on qualsevol desenvolupador pot anar-hi a preguntar coses sobre programació, o a trobar resposta a les seves preguntes, ja que probablement algú les ha preguntat abans, i ja tenen resposta.

Al registrar-se a stackoverflow, es crea un perfil d'usuari. Inicialment aquest perfil no te cap punt, ni insígnia, i moltes de les coses que es poden fer a stackoverflow estaran bloquejades ja que tampoc es tenen els privilegis per a poder fer aquestes accions.

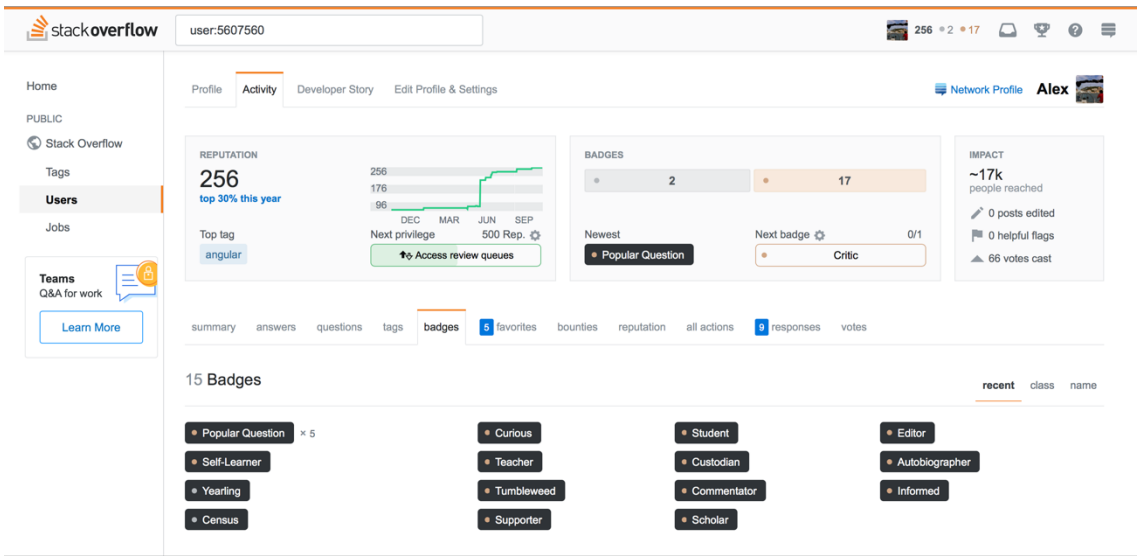
A mesura que l'usuari es va integrant a la comunitat i participa en la web va aconseguint fites i punts. Per a aconseguir punts, s'han de fer bones preguntes que siguin ben valorades per la comunitat, o bé donar respostes correctes, valides i entenedores a les preguntes d'altres usuaris, i que aquestes siguin acceptades i ben valorades per la comunitat.

Amb la consecució de punts, l'usuari obté privilegis per realitzar accions que abans no podia fer (Il·lustració 2). Per exemple, un usuari amb menys d'un punt no pot crear noves preguntes, amb menys de 15 punts no pot votar positivament cap pregunta o resposta, amb menys de 125 punts no pot votar negativament cap pregunta o resposta, etc. A la il·lustració 2 podem veure un llistat d'alguns dels privilegis que s'obtenen amb els punts associats a l'usuari.



Il·lustració 2: relació de punts i privilegis a Stackoverflow

La realització de fites i els punts aconseguits es poden veure resumits al perfil d'usuari (il·lustració 3) junt amb d'altre informació que convida a continuar participant i millorant el perfil.



Il·lustració 3: Perfil d'usuari d'Stackoverflow

1.3 Exemple de gamificació en l'entorn educatiu

En Lluis es un professor d'educació física, i percep que els seus alumnes no estan gaire motivats a les seves classes. Molts dels alumnes no s'esforcen per fer els exercicis, i els realitzen amb desgana.

Per tractar de solucionar-ho, en Lluis busca com motivar els seus alumnes, i acaba trobant, per internet, un manual de gamificació. Amb la informació que extreu d'aquest manual, decideix tractar d'aplicar-ho a les seves classes.

Primer cerca quines **recompenses** són els que motivarien més als seus alumnes, i acaba decidint anotar punts per a cada alumne a un llistat, donar una insígnia adhesiva amb el distintiu de guanyador, i el premi més important que és sortir de classe 10 minuts abans.

Un cop decidides les recompenses, determina quins **objectius** han d'assolir els seus alumnes per tal d'aconseguir els premis. En Lluis estableix que donarà 5 punts a tots els estudiant que arribin puntuals a classe, donarà 5 punts als estudiants que estiguin atents a les seves explicacions i siguin obedients, i donarà 5 punts als estudiants que s'esforcin al fer els exercicis. Per tal d'afegir més emoció a les seves classes, en Lluis també decideix fer competicions, fent carreres entre els alumnes i donant punts als que acabin en la primera posició. I també farà tornejos on els estudiants formaran equips i s'enfrontaran entre si en unes eliminatòries fins que un dels equips guanyi. Als estudiants que guanyin les carreres i les competicions decideix que els donarà la insígnia adhesiva de guanyadors. I finalment, creu que el gran premi de sortir 10 minuts abans el guanyarà l'estudiant que arribi abans als 500 punts i hagi guanyat al menys 10 cops la insígnia de guanyador.

En Lluis escriu en un full les normes del joc, i les explica als estudiants, penjant el full de normes al taulell de la classe. A més, també informa als alumnes que hi haurà una **classificació** que penjarà al taulell al final de cada setmana, en aquesta classificació hi haurà un ranking dels alumnes ordenats de més a menys punts, i en cas d'empat de punts, tenint en compte el numero de vegades que han aconseguit una insígnia de guanyadors. Per acabar d'arrodonir-ho, en el llistat de la classificació, al costat dels punts i les insígnies de cada alumne, escriurà el **percentatge** aconseguit del primer premi de sortir 10 minuts abans de classe. Per tant, un alumne que hagi guanyat 250 punts i hagi aconseguit 5 cops la insígnia de guanyador, haurà assolit el 50% de l'objectiu per a aconseguir el gran premi.

I per evitar que els seus alumnes s'acabin avorrint a les seves classes, en Lluis gradualment anirà **incrementant la dificultat dels exercissis** cada setmana augmentant el número de repeticions dels exercicis, o disminuint el temps per a realitzar alguns dels exercicis. Per exemple, si la primera setmana demana 10 voltes al pati corrent, la segona setmana demanarà 10, i la tercera 15, i així anirà pujant fins a 30.

Al final del trimestre, en Lluis compara els resultats dels seus estudiants amb els del trimestre anterior on no hi va aplicar cap sistema de gamificació, i comprova

que tots els estudiants han pujat la nota com a mínim mig punt, i gairebé tots els estudiants s'han esforçat molt més a l'hora de fer els exercissis, fins el punt que alguns estudiants que no podien fer 20 voltes al pati corrent en el trimestre anterior ara en feien 25, i un d'ells encara havia arribat a completar-les aconseguint la insígnia de guanyador. La sorpresa més gran per al professor va ser que l'Albert, un estudiant que el trimestre anterior es negava a fer molts exercissis i que molts cops arribava tard o es saltava les classes, va fer tot el possible per aconseguir el premi per a poder sortir 10 minuts abans de classe, i de fet va ser el primer en guanyar aquest premi. Aquesta motivació de l'Albert per aconseguir sortir abans de classe, va fer que passes de treure un 3 en el trimestre anterior a treure un excel·lent.

1.4 Aplicacions mòbils com a eina d'aprenentatge

Hi ha moltes formes d'apropar entorns de jocs als estudiants, però un dels mètodes que s'està tornant més atractiu es el de les aplicacions mòbils. En els darrers anys, hi ha hagut una autèntica revolució en el segment de la telefonia mòbil. El nombre de terminals amb capacitat per a connectar-se a internet i executar aplicacions ha augmentat exponencialment, i a més, els preus s'han reduït dràsticament en les games més baixes. Com a conseqüència, gairebé tothom disposa d'un terminal smartphone amb accés a Play Store o App Store. Aquest fet, fa que sigui molt fàcil i atractiu introduir en entorns educatius mètodes de gamificació mitjançant aplicacions mòbils, ja que els mòbils estan a l'abast de tothom. I la utilització d'aquests tipus de tecnologies atreuen i motiven molt a la gent jove.

Un dels objectius d'aquest projecte es el d'obtenir una aplicació mòbil que utilitzi les eines de la gamificació i sigui fàcilment utilitzable tant per mestres, com per alumnes.

En l'actualitat ja existeix tota una gama de aplicacions que, d'una forma o d'un altre, introdueixen elements de gamificació utilitzant aplicacions mòbils. Aquí hi ha un recull d'algunes d'aquestes eines:

- Brainscape
- KnowRe
- Kahoot!
- CodeCombat
- Classdojo
- Classbadges

Brainscape

Il·lustració 4: logotip de Brainscape

Aquesta solució es composa d'un dashboard web i una aplicació mòbil per a iOS/Android. El funcionament es molt senzill, el professor crea jocs de 'flashcards' per que després els alumnes juguin amb ells i els resolguin. A més, el professor també pot encomanar als seus alumnes fer les seves pròpies 'flashcards' per a compartir amb la resta de la classe.

Les 'flashcards', o targes mnemotècniques, son targes on s'escriuen preguntes per un costat i les respostes per l'altre costat. Aquestes targes s'utilitzen com a mètode d'aprenentatge per ajudar a la memorització en base a la repetició espaiada.

El sistema no acaba aquí, les targes creades es poden compartir, i de fet ja hi existeix tot un repositori gratuït de 'flashcards', organitzades per matèries, compartides per altres usuaris. I també hi existeix un repositori de 'flashcards Premium' de pagament. Amb aquest sistema el professor pot començar a utilitzar l'aplicació utilitzant les 'flashcards' ja existents, o fer-ne les seves pròpies.

KnowRe

Il·lustració 5: logotip de Knowre

knowre es una plataforma orientada a l'ensenyament de las matemàtiques, i te un plantejament molt original. Presenta al estudiant com a un avatar en un mapa dividit per zones, o nivells. Tret de la primera zona, la resta de zones estan bloquejades, i només es poden desbloquejar superant les zones prèvies. Quan l'estudiant entra a una zona se li mostra un còmic que planteja un problema que haurà de solucionar per superar la zona.

El mestre disposa d'un dashboard on pot veure els avenços de cada un dels estudiants, per a poder fer un tipus d'ensenyament més personalitzat a cada alumne.

Kahoot!

Il·lustració 6: logotip de Kahoot!

Kahoot es una plataforma que permet fer jocs de preguntes i respostes. Ofereix diversos tipus de jocs com joc de preguntes, enquesta, debat o 'jumble'. Tots el modes de joc son similars, però cadascun amb les seves particularitats.

El professor te accés a una taula amb els participants dels jocs on podrà veure de forma clara i ràpida qui ha participat en el concurs, i que ha respost cada participant a cada pregunta.

Kahoot també disposa d'un repositori de jocs de preguntes a les que el professor hi tindrà accés i podrà utilitzar sense haver de fer les seves pròpies, i on també hi podrà compartir les que ha fet.

Kahoot disposa de diverses modalitats de pagament, on s'incorporen algunes funcionalitats extra segons el pla triat.

CodeCombat

Il·lustració 7: logotip de CodeCombat

Aquesta es un plataforma d'ensenyament de tècniques de programació, que converteixen les classes en un autèntic videojoc. Aquí els estudiants hauran de programar el moviments i accions dels personatges del videojoc. Com millor sigui el codi dels alumnes, més fàcil els serà guanyar nivells i avançar en el joc.

El millor de tot es que els estudiants escriuen el codi en temps real i veuen en el moment el resultat del codi que han escrit, i això converteix alguns conceptes una mica abstractes del codi en accions concretes i visibles.

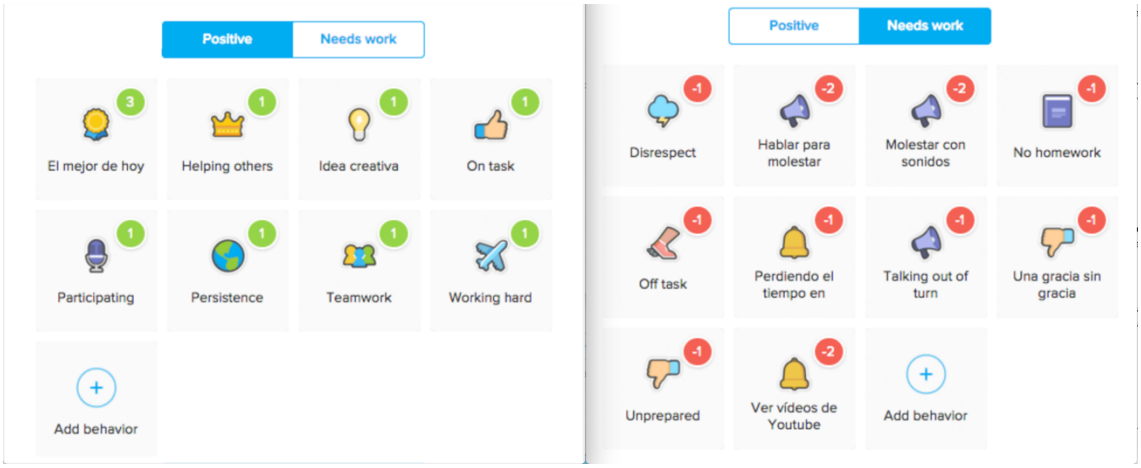
Els cursos de CodeCombat estan disponibles en Javascript i Python.

Classdojo

Il·lustració 8: logotip de Classdojo

Aquesta aplicació tracta de motivar als estudiants amb un sistema de puntuació. De fet, l'objectiu d'aquesta aplicació es permetre al professor puntuar el comportament dels seus estudiants (Il·lustració 9), atorgant una puntuació positiva als alumnes que facin un bo treball, i una puntuació negativa quan tinguin comportaments poc adequats.

Les puntuacions s'adjudiquen als alumnes mitjançant insígnies personalitzables que duen atribuïda una quantitat de punts que sumar o restar a l'alumne.



Il·lustració 9: insígnies i puntuacions assignades

La aplicació també permet realitzar un control d'assistència dels alumnes. I també té una funcionalitat que es diu 'Class Story', que es semblant al mur de Facebook, i on es poden penjar textos i imatges per mostrar el treball que s'ha anat realitzant a classe.

A part dels professors i els alumnes, les famílies dels estudiants també poden participar a Classdojo mitjançant missatges privats amb el professor, veient el 'Class Story', o veient els progressos de l'estudiant.

Classbadges

Il·lustració 10: logotip de Classbadges

Classbadges es una aplicació que permet al professor atorgar als seus estudiants insígnies o premis pel ser bons estudiants o per bon comportament.

1.5 Conclusió sobre Classpip i les aplicacions de gamificació

Un cop explicat que es la gamificació, per a que serveix, com s'utilitza a stackoverflow, com es podria utilitzar en un entorn educatiu. S'ha procedit a aportar exemples on la gamificació s'ha introduït a l'entorn educatiu mitjançant aplicacions tecnològiques. Concretament, en el punt anterior s'han llistat i explicat diverses aplicacions de gamificació per a l'entorn educatiu.

En els exemples d'aplicacions, s'ha vist com n'hi ha aplicacions per a atorgar punts i insígnies, aplicacions per atorgar nomes insígnies, aplicacions per fer preguntes i respostes, etc.

Però no hi ha ni una sola aplicació de gamificació que permeti assignar punts i insígnies, fer qüestionaris, rankings d'estudiants, grups, tornejos, col·leccions de cromos i fer treballar totes aquestes eines transversalment connectant-les. Fent que, per exemple, es puguin fer tornejos de preguntes en grups d'estudiants, i donar punts i una insígnia a tots els membres del grup guanyador del torneig, així com tres sobres de cromos d'una col·lecció al grup guanyador, dos sobres al segon grup i un sobre al tercer grup.

Per tant, Classpip es una aplicació que, tot i que es basa en eines ja existents, es molt innovadora, ja que integra tota una sèrie d'elements que fins ara no s'han vist integrats en una sola aplicació. I aquest es el principal motiu de la creació de Classpip, crear un entorn que utilitzi molts dels jocs possibles, i els integri en una única aplicació.

Aquest projecte va néixer inicialment com un projecte de fi de Master, però després de la participació de diversos alumnes ha crescut molt, incorporant noves funcionalitats en cada un dels projectes i aconseguint una eina de gamificació molt complerta.

En el següent capítol s'oferiran detalls sobre les tecnologies que s'ha utilitzat per a realitzar el projecte, i les diferents funcionalitats que s'han anat afegint amb el temps.

CAPÍTOL 2. INICIACIÓ AL PROJECTE

Un cop introduït el concepte de gamificació, les avantatges de motivació que aporta en un entorn educatiu, i alguns exemples de com funciona i com introduir-ho a l'aula mitjançant aplicacions, s'explicarà la motivació d'aquest projecte, quins elements el comporten, quines funcionalitats se li han afegit, i quins son els objectius concrets d'aquest projecte.

2.1 Motivació

L'any 2016 el Departament d'Arquitectura de Computadors de la Universitat Politècnica de Catalunya comença un projecte de final de Màster amb un alumne per desenvolupar Classpip, centrat inicialment en la seva arquitectura.

Uns anys després i amb la participació de diversos alumnes amb els seus respectius treballs de final de grau, el projecte ha crescut amb noves funcionalitats i jocs. Després del darrer projecte, s'han unit les funcionalitats desenvolupades en els diversos projectes existents fins ara en un únic projecte, i s'han creat relacions entre elles de forma que puguin interactuar entre si.

L'enorme creixement del projecte inicial, i el fet de que cada projecte s'hagi plantejat individualment han provocat que la documentació i el codi del projecte estiguin molt dispersos i sigui una mica complicat per als nous estudiants introduir-se al projecte.

La meva motivació es la de facilitar la incorporació de nous estudiants preparant un manual d'iniciació on hi sigui tota la documentació del projecte ordenada i fàcil de trobar i entendre. A més també vull millorar la forma de treballar per a que sigui molt fàcil integrar els treballs de cada estudiant al projecte i per acabar m'agradaria millorar alguns aspectes del codi, el disseny i de la usabilitat de les aplicacions. De manera que, en acabar el meu projecte, a curt termini es puguin començar a fer proves reals amb professors i alumnes.

2.2 Descripció de l'aplicació

L'aplicació de Classpip esta conformada per tres elements principals:

- una aplicació web anomenada dashboard.
- Una aplicació mòbil per a Android i iPhone.
- una API que proporciona el servei de dades al dashboard i a les aplicacions mòbils.

En els següents punts quedaran explicades les tecnologies emprades en les tres parts de l'aplicació i les dependències que hi entre aquestes, a més d'una breu explicació de les funcionalitats desenvolupades en els projectes anteriors.

2.3 Llistat de funcionalitats

L'aplicació funciona amb una sèrie de funcionalitats que s'han desenvolupat prèviament en projectes separats, i que s'han integrat en un únic projecte per a poder funcionar conjuntament. Fins i tot s'han creat algunes associacions entre aquestes funcions per tal de que puguin interactuar entre elles. Aquest mòduls són:

- **Col·leccions**
  Aquesta funcionalitat permet crear col·leccions de cartes col·leccionables. Els alumnes podran jugar a aquesta funcionalitat aconseguint **recompenses** de paquets de cromos quan acompleixin els **objectius** plantejats pel professor amb la finalitat de completar tot el **percentatge** de les col·leccions. Aquesta funcionalitat permet crear, esborrar i assignar cartes als estudiants. També es poden generar sobres que contenen diverses cartes aleatòries.

- **Punts i insígnies**
  Amb aquesta funcionalitat el professor serà capaç de crear insígnies que s'assignaran als estudiants que compleixin certs requisits. I també proporciona al professor la possibilitat de crear punts que després podrà atorgar als estudiants. Aquest joc proporciona motivació als estudiants al poder aconseguir **recompenses** amb l'assoliment d'**objectius** i complint amb les normes plantejades pel professor.

- **Qüestionaris i preguntes**
  En aquesta funcionalitat, el professor podrà crear qüestionaris que més tard completaran els estudiants mitjançant l'aplicació o el dashboard. Presumiblement, i a discreció del professor, el **nivell de dificultat** dels qüestionaris hauria de pujar progressivament segons avanci el curs, ja que generalment el coneixement de les assignatures es acumulatiu.

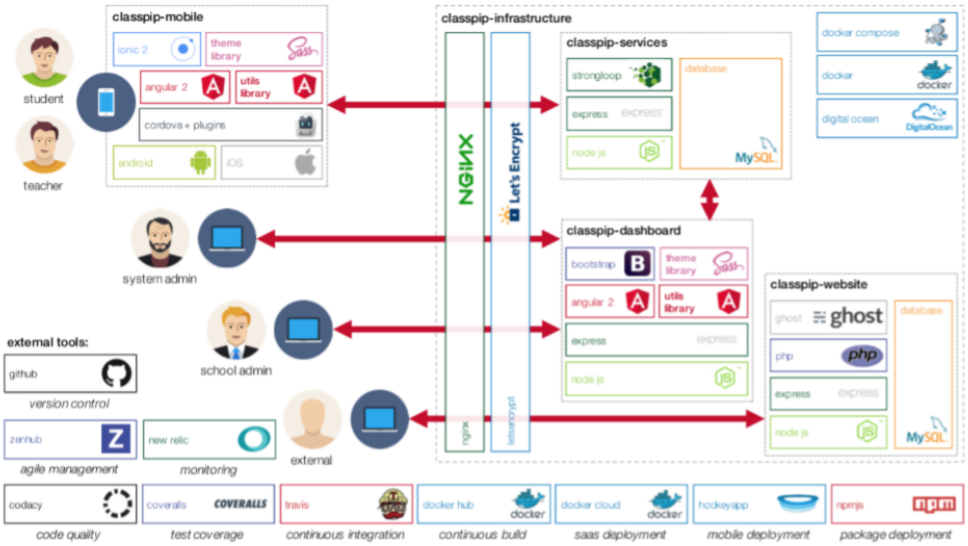
- **Competicions**
  Amb la funcionalitat de competicions el poden crear tornejos per a que els estudiants es puguin enfrontar entre ells i passar a la següent fase si guanyen. D'aquesta manera es crea la possibilitat de fer una **classificació** d'estudiants amb els resultats de les competicions.

- **Grups**
  Aquesta funcionalitat ofereix la possibilitat de formar grups d'estudiants per a que puguin fer treballs en grup o qualsevol altre cosa que requereixi el professor.

- **Interaccions entre funcionalitats**
  Son associacions realitzades entre les funcionalitats explicades en els punts anteriors. Aquestes relacions permeten que les funcionalitats interactuïn entre elles fent que, per exemple, els estudiants puguin guanyar punts quan contesten correctament els qüestionaris, o fer una competició entre grups d'estudiants on els guanyadors de la competició guanyin un sobre de cromos i una insígnia distintiva.

2.4 Arquitectura

Les tecnologies emprades en aquest projecte son les mateixes que les que hi havia inicialment (Il·lustració 11), amb l'excepció de bootstrap que ha sigut substituït per Angular Material. A més, el website de projecció exterior ja no esta fet amb php, sinó que esta fet amb Angular 7.2.1 i Angular Material 7.2.1.



Il·lustració 11: arquitectura i estructura de l'app

Classpip es un projecte format per quatre repositoris. Cada un d'aquests repositoris te una finalitat diferent. Els repositoris son els següents:

Classpip-service

Es tracta d'una API REST que connecta amb la base de dades, i que ofereix els endpoints per a poder accedir a aquesta des de l'exterior. L'avantatge d'utilitzar una API REST es que es independent de les altres parts del projecte i els llenguatges de programació que utilitzin cada un d'ells.

Explicat de forma molt senzilla i simplificada, els endpoints d'una API REST, son direccions html amb les quals es pot accedir a llegir, escriure, modificar o esborrar dades de la base de dades. Les peticions http que es generen mitjançant aquestes direccions, contenen tota la informació necessària per a realitzar les connexions entre les aplicacions i l'API. La part del dashboard i de l'aplicació mòbil utilitzen aquests endpoints de l'API per a poder obtenir i enregistrar les dades.

L'API REST s'ha creat utilitzant Strongloop, que a més de generar els endpoints per a realitzar les connexions, també disposa d'un panell web des d'on veure i utilitzar els endpoints disponibles. La base de dades utilitzada al projecte es mysql i l'eina que proporciona aquests serveis es Strongloop, que funciona sobre express i node.js.

Classpip-dashboard

Aquest es l'accés web a l'aplicació. Serveix tant per a l'administrador de l'escola, com per a professors o estudiants. Aquí estan disponibles tots els serveis de Classpip per a tots els seus usuaris.

Les tecnologies d'aquesta eina son una base de node.js i express sobre la que funciona Angular 4.4 i Angular Material 2.0.0.

Classpip-mobile

Es tracta d'una aplicació per a mòbils iOS i Android d'es d'on poden operar tant els professors com els estudiants.

Aquesta part funciona amb gairebé el mateix codi d'Angular 4.4 i Angular material que el dashboard, i funciona sobre Ionic 2. Ionic es un framework que utilitza Apache Cordova per a, d'entre d'altres coses, proporcionar accés als components natius del telèfon.

D'aquesta manera, encara que en aquest projecte no s'utilitza, es podria tenir accés a elements del telèfon com el GPS, la càmera, l'acceleròmetre, el lector d'empremtes, etc. I s'obre la porta a futurs desenvolupaments com ara, per exemple, el poder loginar-se a l'aplicació mitjançant el lector d'empremtes, o poder pujar fotografies fetes amb la càmera del mòbil per a les preguntes, etc.

Classpip-onboarding

Aquest repositori es nou, i el seu desenvolupament s'ha dut a terme amb aquest projecte.

Il·lustració 12: estructura web d'onboarding

Encara que el repositori s'ha anomenat onboarding, aquesta pagina web també proporciona projecció externa, on a més de donar visibilitat al projecte, proporcionarà tot el material per a poder iniciar-se al desenvolupament del projecte, i el material que es generi per a la utilització del dashboard i l'aplicació mòbil.

Aquesta pagina s'ha desenvolupat utilitzant l'ultima versió d'Angular i Angular Material, que funcionen sobre Express i Node.js (il·lustració 12). Es va iniciar amb versions inferiors d'angular i material, i s'ha anat actualitzant a versions més actuals a mesura que han anat publicant-se.

La pàgina web d'aquest projecte s'ha posat a funcionar en un servidor gratuït d'Heroku. Addicionalment, s'ha creat una connexió entre el repositori de Github

i Heroku, de forma que quan es pugi codi a la branca màster del repositori de Github es publiquin els canvis automàticament a Heroku.

2.5 Objectius

Classpip es un projecte que ha crescut molt des dels seus inicis, encara que això no significa que encara no hagi obertes línies de millora del projecte. En base a aquestes possibilitats de millora es procedeix a declarar uns objectius a complir en aquest Treball de Fi de Grau.

Al començar aquest treball, es va fer evident un problema, i es que tota la documentació del projecte estava absolutament fragmentada en diversos arxius PDF, diferents repositoris, cadascun d'ells amb diferents funcionalitats del projecte desenvolupades per separat. Per aquesta raó es feia molt difícil introduir-se al projecte a l'haver de recopilar dades de moltes fonts diferents. I d'aquesta problemàtica va néixer l'objectiu de recopilar tot aquest material i ajuntar-ho de forma estructurada en una única pàgina web. Pàgina que a més ha de tenir una landing per a utilitzar com a projecció externa.

Per tal de minimitzar aquesta fragmentació en el futur, també es vol dissenyar un flux de treball amb git que faciliti la integració dels Treballs de Final de Grau realitzats per altres estudiants, i que serà inclòs com a manual en la nova web de projecció externa i material d'onboarding. Al mateix temps, també es realitzarà, i s'inclourà a l'apartat de material d'onboarding, un manual que faciliti el desenvolupament i manteniment d'aquesta mateixa pàgina web a futurs estudiants per tal de que puguin continuar actualitzant els materials d'onboarding de la web i aquests no quedin obsolets i perdin la seva utilitat.

Addicionalment, es vol desenvolupar una funcionalitat de nivells i rangs, que serveixi com a complement de la funcionalitat de punts. Aquest sistema de rangs es basarà en donar com a **recompensa** uns títols i unes imatges assignades als estudiants quan arribin a complir uns **objectius** de quantitat de punts, i que serviran per a destacar sobre d'altres estudiants. Al mateix temps els punts obtinguts pels alumnes s'utilitzaran per a aconseguir un nivell, que també pot servir per a fer una **classificació**.

Un altre dels objectius es crear una funcionalitat que guardi les **recompenses** obtingudes pels estudiants de forma que després l'aplicació les pugui recuperar de forma senzilla i ràpida. Aquesta funcionalitat nova, que s'anomenarà Rewards, també s'utilitzarà a l'hora de fer rankings i **classificacions** d'estudiants, ja sigui pels seus nivells, pels punts que han aconseguit, o per altre tipus de recompensa que es recuperi utilitzant aquesta funcionalitat.

També s'ha plantejat que la pagina d'inici del dashboard s'ha de redissenyar i afegir funcionalitats, aprofitant els nous desenvolupaments de les funcionalitats de rangs i nivells i el de Rewards. Afegint totes les funcionalitats de l'administrador al dashboard, ja que fins ara, les dades utilitzades al projecte eren dades preparades en fitxers que es creaven al iniciar la part de serveis, i no hi havia cap manera de crear col·legis, professors, estudiants, etc.

Altre propòsit es el de trobar parts del codi que siguin susceptibles de millores i realitzar aquestes millores, buscant també millores que es puguin realitzar en el aspecte de la usabilitat i que millorin l'experiència de l'usuari al utilitzar el dashboard.

Finalment ha de quedar un projecte en el que es puguin arribar a fer les primeres proves en un entorn real per poder extreure'n millor les seves virtuts i carències que en un entorn de desenvolupament no s'arriben a detectar.

CAPÍTOL 3. NOUS DESENVOLUPAMENTS

Un dels objectius d'aquest projecte es afegir algunes funcionalitats noves. Com es va comentar al capítol anterior, es realitzarà una nova funcionalitat que **recompensi** els estudiants amb un títol i un nivell quan assoleixin els seus **objectius** de punts. I també es realitzarà una funcionalitat anomenada Rewards, destinada a facilitar la recuperació de les dades de les **recompenses** obtingudes pels estudiants, i la elaboració de **classificacions** d'estudiants amb aquestes dades.

3.1 Funcionalitat de rangs i nivells

Aquest mòdul, funciona com a adició o complement al mòdul de punts. Fent que sigui molt més emocionant per als estudiants aconseguir punts. Aquest mòdul ofereix unes **recompenses** al atorgar un nivell i un rang superior al aconseguir els **objectius** d'una quantitat de punts concreta fixada pel professor. D'aquesta manera, no només es dona una utilitat als punts aconseguits pels estudiants, sinó que a més se'ls en proporciona un **objectiu** assolible una vegada reunida una certa quantitat de punts.

Amb els objectius d'aconseguir un nivell més alt, i un rang més alt que els dels companys, es genera una motivació extra en els estudiants que els impulsarà a voler aconseguir més punts.

3.1.1 Funcionalitat de rangs i nivells a Services

Per a fer el mòdul de rangs i nivells, es comença per realitzar certes modificacions a l'element de Services. S'ha de començar per declarar Range i les seves propietats.

Les propietats utilitzades a rangs son les següents:

```
"properties": {
    "nombreRango": {
      "type": "string",
      "required": true
    },
    "puntosRango": {
      "type": "number",
      "required": true
    },
    "imageRangoLink": {
      "type": "string",
      "required": true
    },
    "altImageRangoLink": {
      "type": "string",
      "required": false
    }
}
```

nombreRango es el nom que se li vol donar al rang, puntosRango es el mínim número de punts necessari per a assolir aquest rang, imageRangoLink es la imatge que es mostrarà quan els usuaris arribin a aquest rang, i altImageRangoLink actualment no s'utilitza, però està en previsió de fer servir una imatge alternativa per al rang en algunes situacions.
La relació de Range amb School es Range belongsTo School amb el foreginKey schoolId. I School hasMany Ranges amb foreignKey SchoolId. El que vol dir que una escola pot tenir molts rangs, però un rang només pot pertànyer a una escola.

Per a completar la relació també s'ha d'editar la declaració d'school per afegir-la.

Finalment s'ha d'agregar al model-config.json la configuració:

```
"Range": {
    "dataSource": "db",
    "public": true
  }
```

Un cop fet això s'omplen les dades amb un arxiu create-ranks.js on s'hi afegeixen alguns rangs d'exemple que estaran disponibles després d'iniciar l'API rest.

3.1.2 Funcionalitat de rangs i nivells a Dashboard

Model

Per implementar aquest mòdul en el dashboard s'ha començat per crear el model del rang. El model es defineix a l'arxiu src/app/shared/models/rango.ts. Es en aquest arxiu on es declaren totes les propietats del rang, el constructor, les funcions toObject i toObjectArray i els getters i els setters.

La funció toObject serveix per a traduir la resposta de l'API i convertir-la en un objecte Rango amb el que Angular pot treballar molt fàcilment.

La funció toObjectArray, converteix la resposta d'un llistat de rangs en un array d'objectes Rango.

Amb els getters i els setters, les propietats d'aquest objecte Rango son accessibles a lectura i a escriptura.

Ara només queda afegir la exportació del model a l'arxiu index.ts dels models situat a la mateixa carpeta:

```
export * from './rango';
```

Servei

Per a utilitzar aquest mòdul es va crear el servei de level.service.ts. En aquest servei hi ha una funció que calcula el nivell de l'estudiant mitjançant el seus punts. En un principi, la forma de calcular el nivell es molt senzilla, només cal dividir per 10 el total de punts i arrodonir el resultat cap avall. La funció resultant es la següent:

```
public getLevel(points: Number): Number {
    if (points <= 0) {
      points = 1;
    }
    const level = Math.floor(points.valueOf() / 10);
    return level;
}
```

Per a completar el mòdul de rangs només resta poder obtenir el rang de l'estudiant a partir dels seus punts. Per a realitzar aquesta tasca, es demanen el rang a l'API i a la resposta es busca quin es el rang més gran on el seus punts requerits siguin mes petits o iguals que els punts de l'estudiant. Com a resposta, aquesta funció retorna el rang que obtindria aquest estudiant. La funció es la següent:

```
public getRank(points: Number): Rango {

    const options = this.utilsService.getOptions();

    if (points <= 0 || points === undefined) {
      points = 1;
    }

    this.http.get(AppConfig.RANGE_URL, options)
      .map((res: Response) => res.json())
      .subscribe(
        (response) => {
          const max = response.reduce(
            function (i, j) {
              return j.puntosRango <= points ? Math.max(i,
j.puntosRango) : i;
            }, Number.MIN_VALUE
          );
          this.rank = response.filter(rango =>
rango.puntosRango >= max).pop();
        });

    return this.rank;
}
```

Hi ha una funció per a obtenir tot el llistat de rangs, inicialment es va crear una funció d'ordenació que ordenava l'array de rangs per la propietat puntosRango. Però un cop descoberta la possibilitat d'aplicar filtres a les peticions de l'API, la funció d'ordenació va perdre tot el sentit, ja que utilitzant el filtres, les dades ja venen amb l'ordre desitjat i no cal tractarles per a canviar aquest ordre. La funció de llistar els rangs es la següent:

```
public getAllRanks(): Observable<Rango[]> {

    const options = this.utilsService.getOptions();
    const url = AppConfig.RANGE_URL +
'?filter[order]=puntosRango%20ASC';

    return this.http.get(url, options)
      .map((response: Response) =>
Rango.toObjectArray(response.json()));
  }
```

Per acabar, aquest servei ofereix les funcions de crear un rang nou, editar un rang existent i esborrar un rang. D'aquesta manera, s'ofereixen les eines per a poder gestionar els rangs.

3.2 Funcionalitat de Rewards

Per la forma en la que es va dissenyar l'aplicació, era molt costós aconseguir el total de punts d'un estudiant.

La raó d'això es que a l'estudiant s'assignen "objectes punt", que contenen tota la informació sobre el punt atorgat a l'estudiant. I quan es vol aconseguir el total de punts, s'ha de demanar a l'API les relacions de punts de l'estudiant, després s'han de demanar els objectes punts que l'estudiant te assignats i finalment s'han de recórrer tots els objectes punts sumant el valor de cada un d'ells.

Aquest funcionament es correcte, però no gens escalable a una gran quantitat d'estudiants, ja que genera massa peticions i massa processament de dades per a obtenir la xifra de punts que hauria de ser molt fàcil d'aconseguir.

Fer algunes representacions de relacions d'estudiants i punts es converteix en una tasca bastant complexa.

De la mateixa manera, les relacions entre els estudiants i els rangs i els nivells seria una mica complexa en tractar-ho de fer amb una gran quantitat d'estudiants alhora.

Es per aquesta raó, que es crea el mòdul de Rewards, que no es més que una relació ja feta i guardada amb la resta de dades dels punts totals de l'estudiant, el seu rang, el seu nivell, els punts necessaris per arribar al següent nivell, i els identificadors dels punts de l'estudiant amb la quantitat de cada un d'ells que se li ha assignat.

3.2.1 Funcionalitat de Rewards a Services

Per a fer el mòdul de Rewards, es comença per realitzar certes modificacions a l'element de Services. S'ha de començar per declarar Reward i les seves propietats.

Les propietats utilitzades a Rewards son les següents:

```
"properties": {
    "points": {
      "type": "number"
    },
    "points_obj": {
      "type": "string"
    },
    "badges_obj": {
      "type": "string"
    },
    "level": {
      "type": "number"
    },
    "next_level_points": {
      "type": "number"
    },
    "rank": {
      "type": "string"
    }
  },
```

La propietat points son els punts totals que ha obtingut l'estudiant, la propietat points_obj es un array que s'ha convertit a string i que guarda una relació entre els identificadors dels objectes dels punts i la quantitat de vegades que aquests s'han assignat als estudiants, badges_obj no s'utilitza actualment, però es una propietat molt similar a la dels punts que s'utilitzarà en pròxims projectes, level pertany al mòdul de rangs i nivells , i es el nivell que ha assolit l'estudiant amb els seus punts, next_level_points son els punts que li manquen a l'estudiant per a aconseguir el següent nivell, i rank es el rang que té actualment l'estudiant provinent també del mòdul de rangs i nivells.

La relació de Reward amb Student es de belongsTo, utilitzant el foreign key d'studentId, al mateix temps, la relació de Student amb Reward es de hasOne, amb el foreign key de l'identificador del Reward. Es a dir, tot estudiant te un Reward, i tot Reward pertany a un estudiant.

Per a poder completar la relació entre Reward i Student, també s'ha d'editar la definició d'estudiant afegint la relació hasOne amb Rewards.

Finalment s'ha d'agregar al model-config.json la configuració:

```
"Reward": {
  "dataSource": "db",
  "public": true
},
```

Un cop fet això s'omplen les dades amb un arxiu create-rewards.js on s'hi afegeixen els rewards dels estudiants d'exemple que estaran disponibles després d'iniciar l'API rest.

Ara només queda afegir la exportació del model a l'arxiu index.ts dels models situat a la mateixa carpeta:

```
export * from './rango';
```

3.2.2 Funcionalitat de Rewards al Dashboard

Model

El primer pas es declarar el model dels Rewards en el dashboard. El model es defineix a l'arxiu src/app/shared/models/reward.ts. Es en aquest arxiu on es declaren totes les propietats del Reward, el constructor, les funcions toObject i toObjectArray i els getters i els setters.

La funció toObject serveix per a traduir la resposta de l'API i convertir-la en un objecte Reward amb el que Angular pot treballar molt fàcilment.

La funció toObjectArray, converteix la resposta d'un llistat de Rewards en un array d'objectes Reward.

Amb els getters i els setters, les propietats d'aquest objecte Reward son accessibles a lectura i a escriptura.

Ara només queda afegir la exportació del model a l'arxiu index.ts dels models situat a la mateixa carpeta:

```
export * from './reward;
```

Servei

Per a poder utilitzar el mòdul de Rewards s'ha generat un servei amb el nom de rewards.service.ts. En aquest servei hi ha múltiples funcions que s'encarreguen de sumar els punts als estudiants quan aquests reben nous punts, obtenir el nou nivell, calcular els punts que queden per arribar al següent nivell, obtenir el rang de l'estudiant si aquest ha assolit un de nou, i enregistrar quin es l'objecte punt que se li ha atorgat a l'estudiant i quants en te en total d'aquest tipus.

Tot comença amb la funció addPointsProcess(), que obté un llistat dels rangs. Un cop que la funció rep els rangs, es procedeix a comprovar que l'estudiant en qüestió te un Reward assignat. En principi, tots els estudiants han de tenir a la força un Reward assignat, però en el cas de que per alguna raó, un estudiant no en tingués, tot el procés donaria un error. Per aquesta raó es fa aquesta

comprovació, i si l'estudiant no te un Reward, es crea un de nou buit i llavors es continua amb el procés sumant-li els punts que l'han iniciat i els seus passos consecutius.

En el cas de que, com hauria de ser sempre, l'estudiant tingui un Reward assignat es procedirà a sumar el punt i la resta del procés de manera normal.

El codi per a comprovar l'existència del Reward es el següent:

```
public rewardCheck(studentId: string, pointId: string, value:
number) {
    this.checkIfRewardExists(studentId).subscribe(
      (res: Array<boolean>) => {
        this.exists = res['exists'];
        if (this.exists) {
          this.addPoint(studentId, pointId, value);
        } else {
          this.addPointToNewReward(studentId, pointId,
value);
        }

      }
    );
  }

  public checkIfRewardExists(studentId: string):
Observable<Array<boolean>> {
    const options = this.utilsService.getOptions();
    const url = AppConfig.ONLY_REWARDS_URL + '/' + studentId
+ AppConfig.EXISTS_URL;
    return this.http.get(url, options)
      .map((res: Response) => res.json());
  }
```

Del procés d'afegir el punt, el més destacable es la funció que afegeix l'id de l'objecte punt i la quantitat total de punts d'aquest id que ha obtingut l'estudiant que es guarden en un array, on el key de l'array es l'identificador del punt, y el value el numero de vegades que l'estudiant ha obtingut el punt. Aquest array s'ha de passar per un "stringify", ja que es el que es necessita per guardar aquesta informació a la base de dades.

A aquesta funció se li passa com a paràmetre l'string de point_obj i el pointId, i retorna un altre string amb el nou point_obj format.

```
private addPointObj(pointObj: string, pointId: string):
string {
    const point_obj = JSON.parse(pointObj);
    if (point_obj[pointId] !== undefined) {
      point_obj[pointId] += 1;
    } else {
      point_obj[pointId] = 1;
    }
    return JSON.stringify(point_obj);
  }
```

Per a completar el servei, també n'hi ha mètodes per a obtenir el Reward d'un estudiant, i per a obtenir un llistat de tots el Rewards. A més, ja que un dels objectius es poder fer estadístiques amb els punts dels estudiants, n'hi ha un mètode que fent ús dels filtres de Strongloop, retorna un llistat de estudiants amb el seu avatar i amb els seus respectius Rewards, ja sigui d'una escola en concret o tots els estudiants de totes les escoles.

```
public getAllStudentsWithRewards(schoolId: string = ''):
Observable<Student[]> {

    const options = this.utilsService.getOptions();
    let request_option =
'?filter=%7B%22include%22%3A%5B%22rewards%22%2C%22avatar%22%5D%7D';

    if (schoolId !== '') {
      request_option =
'?filter=%7B%22include%22%3A%20%5B%22rewards%22%2C%22avatar%2
2%5D%2C%20%22where%22%3A%7B%22schoolId%22%3A' + schoolId +
'%7D%7D';
    }
    const url = AppConfig.STUDENT_URL + request_option;
    return this.http.get(url, options)
      .map((response: Response) =>
Student.toObjectArray(response.json()));
  }
```

Implementació

Existeix molt de treball previ d'altres estudiants en aquest projecte, i la forma de assignar punts als estudiants que s'utilitzava fins ara ja funcionava prou be. Per aquesta raó la forma d'implementar el sumatori de punts de Rewards ha sigut paral·lela a la ja existent.

La forma d'iniciar-se el procés del servei Reward, es a partir de la funció que fins ara assignava punts. Concretament s'inicia en la funció postPointrelation() situada al servei de pointrelation.service.ts. Es en aquest moment, quan executa el següent codi:

```
this.pointService.getPoint(parseInt(pointId, 10)).subscribe(
      (point: Point) => {
        this.rewardService.addPointsProcess(studentId,
point.id, point.value);
      }
    );
```

El fet d'haver utilitzat els filtres include, ha suposat fer canvis en els models de professor, estudiant, etc per a incloure les noves propietats d'avatar, Reward, etc. De no fer-ho, aquestes propietats no son reconegudes i no passen a formar part de les dades amb les que treballarà angular.

CAPÍTOL 4. NOVA PÀGINA WEB

Un cop explicades les noves funcionalitats del projecte, s'explicaran els detalls de la nova pàgina web de projecció externa i material d'onboarding que s'ha desenvolupat des de zero per a cobrir una de les carències de Classpip, que era tenir una documentació centralitzada i ben organitzada que faciliti la incorporació al projecte a nous estudiants.

4.1 Nova web de projecció externa i material d'onboarding

El projecte de Classpip s'ha fet molt gran des de que va començar, gracies a la participació de diversos alumnes. Cada un dels participants del projecte ha generat unes documentacions i uns materials molt interesants, que fins ara estaven dividits en diversos documents i formats. Aquest fet dificultava la introducció de nous estudiants al projecte que havien de recopilar tot el material i posar-lo en comú abans de començar a desenvolupar.

Per aquest fet s'ha estimat necessari desenvolupar una pàgina web que reuneixi tot el material i el presenti d'una forma fàcil de trobar, llegir i seguir sense haver d'estar reunint documents de diverses fonts.

A més, ja s'ha aprofitat aquest desenvolupament també com a web de projecció externa. Fent que la pàgina de home sigui una "landing", on veure informació sobre el projecte Classpip amb imatges grans i frases curtes.

4.2 Tecnologies emprades

Aquesta web per a Classpip s'ha desenvolupat utilitzant les ultimes versions d'Angular i de Material Angular. A més s'ha hagut d'implementar un petit servidor de Node.js amb Express.

El codi s'ha versionat amb l'eina de Git, i s'ha pujat el repositori a Github. La pàgina web s'allotja en els servidors d'Heroku. A més, s'ha creat una connexió entre el repositori de Github i el servidor d'Heroku, que fa desplegaments automàtics de la web a Heroku quan es puja codi nou a la branca master de Github.

La web s'ha dissenyat i desenvolupat pensant en un comportament responsiu que faciliti la navegació des de dispositius mòbils, tabletes i ordinadors. Els estils de la web estan realitzats en sass amb el patró de disseny "mobile first", es a dir, pensant el disseny primerament per a pantalles de dispositius mòbils i després allotjant els components en pantalles de majors dimensions.

També, per a facilitar la lectura del codi de programació, i la possibilitat de copiar-lo fàcilment, s'ha utilitzat el mòdul de node anomenat prismjs, que permet donar un estil de colors visual i entenedor als diversos tipus de codi que es mostren en alguns llocs de la web.

4.3 Seccions

La web esta dividida en tres seccions principals:

- **Home:** Una pàgina “landing” que fa de pàgina d’inici, i que servirà tant per atreure l’atenció sobre la pàgina web com per a fer una breu introducció al projecte amb frases curtes i imatges. El principal objectiu d’aquesta “landing” es servir com a projecció externa del projecte.

- **Usuaris:** Aquesta secció anirà destinada als tres tipus d’usuaris que podran accedir a la web i l’aplicació de Classpip. Contendrà tutorials amb explicacions per a la utilització de la web tant per a professors, com per a estudiants, com per als administradors.

- **Desenvolupadors:** En aquesta secció es on es disposarà tot el material disponible per a facilitar la introducció al projecte com a desenvolupador. D’aquesta manera, nous estudiants que vulguin participar en el projecte de Classpip podran trobar explicacions sobre com preparar l’entorn de desenvolupament, cursos per aprendre a desenvolupar amb angular, Ionic i Strongloop, com utilitzar Git i com utilitzar-lo per treballar en equip, entendre com desenvolupar un petit mòdul al projecte de Classpip, els problemes que es podrien trobar i les seves solucions, i les documentacions dels anterior projectes. Fins i tot, hi ha un manual, sobre com fer modificacions o adicions al mateix projecte de la web d’onboarding (Annex 6.2).

4.4 Desenvolupament

El desenvolupament d’aquest projecte es va començar amb la primera versió d’Angular 6, que aporta unes quantes millores respecte a la versió d’Angular utilitzada en els altres components del projecte.

Un d’aquests avantatges es la facilitat d’instal·lar alguns components, com ara Angular Material amb una senzilla comanda: *“ng add @angular/material”*.

I una de les millors virtuts es la facilitat i rapidesa d’actualització de les versions d’Angular i Material utilitzades també amb una comanda: “ng update”. D’aquesta manera, el projecte s’ha anat actualitzant de versió durant el seu desenvolupament fins a arribar a la versió 7. N’hi ha més detalls sobre l’actualització de versions en el annex de desenvolupament per al projecte d’onboarding.

El codi s’ha estructurat de la mateixa manera que esta estructurada la part del projecte del dashboard, per facilitar la comprensió del codi i la localització de les seves parts.

4.4.1 Menú de navegació

Part del codi s'ha generat aprofitant les avantatges de les ultimes versions d'Angular i de Material. Per exemple, per a generar el header i el menú de navegació es va utilitzar la comanda: *"ng generate @angular/material:materialNav --name shared/navigation"*. Aquesta comanda genera el component de navegació amb una estructura pregenerada que després s'ha hagut de personalitzar. Per fer-la funcionar requereix editar el fitxer app.component.html i substituir el component *<router-outlet></router-outlet>* per *<app-navigation></app-navigation>*. A més també requereix ficar el component router-outlet dins de la plantilla html del component de navegació.

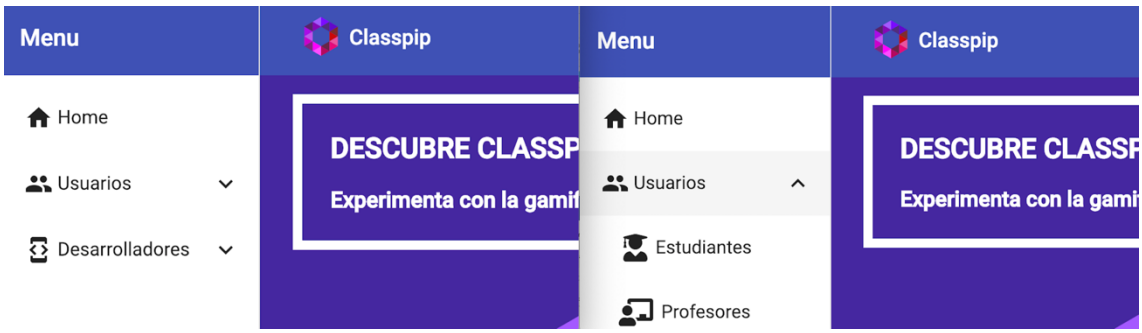
Per a fer les opcions desplegables de dins del menú de navegació, s'ha hagut de generar codi personalitzat parametritzable i reutilitzable.

```
openClose(identifier, caretId) {
      identifier = !identifier;

      var icon = document.getElementById(caretId);
      if(identifier) {
            icon.classList.add('open');
      } else {
            icon.classList.remove('open');
      }
      return identifier;
  }
```

Amb aquesta funció, només cal afegir a l'enllaç pare que l'executi amb un identificador que tindran els seus fills i un identificador per a la fletxa que indica si el menú esta obert o tancat. La crida per al grup de menú d'usuaris queda així: *(click)="usersShow = openClose(usersShow, 'userCaret')"*.

I per acabar el desplegable del menú només resta posar el resultat de la funció openClose a un ngIf en els fills, que serà el que els mostri o amagui, i un altre cop la funció *openclose* també per a que si es fa click sobre un fill es tanqui el menú lateral (Il·lustració 13). Tanmateix, el que s'ha d'afegir al fill es el següent codi: **ngIf="usersShow" (click)="usersShow = openClose(usersShow, 'userCaret')"*.



Il·lustració 13: elements del menú plegats a l'esquerra i desplegats a la dreta

4.4.2 Títols de les pàgines i enllaços

Per qüestions de SEO (Search Engine Optimization) cada pàgina ha de tenir un títol diferent, i aquesta funcionalitat s'ha implementat mitjançant un mètode que ofereix el servei *"Title"*. Això implica utilitzar una funció que canviï el títol de la pàgina en cada un dels enllaços de la web. Nomes cal importar el servei *"Title"* a cada component que tingui enllaços, declarar-ho al constructor i fer servir un mètode que canviï el títol de la pàgina, que en aquest cas s'ha anomenat *"setTitle()"* i el seu codi es el següent:

```
public setTitle( newTitle: string) {
    this.titleService.setTitle( newTitle );
}
```

Després tan sols s'ha de posar la funció de manera que s'activi quan es faci click a l'enllaç. Amb un codi com aquest: *(click)="usersShow = openClose(usersShow, 'userCaret'); setTitle('Classpip Nou títol per a la pàgina');"*

4.4.3 Càrrega de pàgines

El contingut de la pàgina es estàtic, per el que el temps de carrega es menyspreable i l'element de càrrega de la pàgina no serà visible. Però en previsió de que en un futur projecte això canviï i puguin existir temps d'espera abans de poder mostrar el contingut, s'ha creat un mètode de càrrega que mostrarà una barra de carrega fins que els tots components i els seus fills no estiguin llestos.

Per fer això es crea un component que inicia un component de Material anomenat *"mat-progress-bar"* que mostra la barra de progrés de càrrega. I es col·locarà aquest component juntament amb el component "router-outlet" però amb un condicional per a que mentre que el component de la barra de progrés sigui visible, no ho sigui el contingut del router-outlet, i que quan el component del router-outlet sigui visible que no ho sigui la barra de càrrega. Per aconseguir-ho, a la plantilla s'ha de escriure aquest codi:

```
<div [hidden]="!loading" class="loader">
   <h2>Loading...</h2>
   <app-loading></app-loading>
</div>
<div [hidden]="loading" class="router-output">
   <router-outlet></router-outlet>
</div>
```

Per a que tot funcioni, cal utilitzar els *"Lifecycle hooks"*. Els *"Lifecycle hooks"* permeten executar codi quan els components arriben a cert punt del seu cicle de vida. Els punts dels cicles de vida dels components son: quan es creen, es renderitzen, es representen els seus fills, quan canvien les entrades de dades i finalment quan es destrueixen. Per a saber quan s'ha finalitzat la càrrega de la pàgina i es pot mostrar, s'utilitza el ngAfterViewInit, que es crida després de que

un component i els seus fills siguin creats. Es a dir, que es crea quan la vista del component esta completament inicialitzada.

D'aquesta manera es mostrarà la barra de càrrega fins que el component sigui completament visible. Per a aconseguir-ho s'ha de fer servir el següent codi:

```
ngAfterViewInit() {
    this.router.events
      .subscribe((event) => {
      if(event instanceof NavigationStart) {
        this.loading = true;
      }
      else if (
        event instanceof NavigationEnd ||
        event instanceof NavigationCancel
      ) {
        this.loading = false;
      }
    });
  }

}
```

Aquest codi es subscriu als esdeveniments que genera l'enrutador. Concretament posa el valor de "loading" a "true" quan s'inicia la navegació cap a un altre ruta, i finalitza la càrrega quan s'acaba la navegació cap a l'altre ruta o aquesta es cancel·la atorgant a la variable *"loading"* el valor *"false"*.

4.4.4 Visualització del codi als tutorials

La pàgina web d'onboarding reuneix molts tutorials de desenvolupament, alguns d'ells escrits, i d'altres en vídeo. Fins ara el codi dels tutorials es posava amb captures de l'IDE, però les imatges amb text no son bones per a la responsivitat, i no son gaire usables a l'hora de llegir codi, i sobretot de copiar-lo.

La millor solució es utilitzar el codi directament en els tutorials, però si només s'escriu el codi, aquest perd llegibilitat. I la solució per la pèrdua de llegibilitat es formatar el codi amb el mateix indentat i colors que a l'IDE, afegint a més el numero de les línies de codi i un botó que copia tot el codi que es mostra en aquell apartat (Il·lustració 14).

Després d'avaluar diferents formes de solucionar aquest problema, es va optar per utilitzar el mòdul prism.js, que automatitza el formatat del codi, afegeix els números de línia i el botó per a poder copiar el codi.

La instal·lació d'aquest mòdul i la creació del servei necessari per a poder utilitzar Prism.js es pot llegir detalladament en l'annex del Manual de desenvolupament de la web d'onboarding, a l'apartat Crear un nou servei.

Il·lustració 14: formatat de codi a la pàgina de onboarding utilitzant Prism.js

Al mateix annex, a l'apartat d'inserir fragments de codi hi ha informació sobre com utilitzar Prism.js per a mostrar el codi formatat, els números de línia, i les diverses opcions que ofereix aquest mòdul.

4.4.5 Drag & Drop

Les pàgines on hi ha videotutorials, s'han fet d'una forma diferent a les altres per a poder implementar la funció d'arrossegar i deixar anar (Drag & Drop). Aquesta funció consisteix en poder agafar amb el ratolí una targeta amb un vídeo i arrossegar-la per la pantalla fins a un altre posició al llistat, lloc on es deixa anar per a que aquesta targeta passi a ocupar la nova posició, podent així ordenar els vídeos de la forma més convenient a l'usuari.

Per aquesta raó, aquestes pàgines utilitzen un model on s'han declarat un títol i un enllaç de Youtube, i un fitxer amb les dades. Aquest es el model declarat:

```
export class DataCard {
  title: string;
  videoLink: string;
}
```

Per a implementar la funció de Drag & Drop s'ha de realitzar la següent importació:

```
import {CdkDragDrop, moveItemInArray} from
'@angular/cdk/drag-drop';
```

I per últim s'ha d'escriure la funció:

```
drop(event: CdkDragDrop<DataCard[]>) {
    moveItemInArray(this.cards, event.previousIndex,
event.currentIndex);
}
```

On *DataCard[]* es la declaració de que es tracta d'un array del model declarat inicialment i *this.cards* es l'array amb les dades importades des de el fitxer.

A la plantilla s'ha d'afegir el tag *cdkDrag* a les targetes, que les permet arrossegar. I el tag *cdkdropList* al llistat per a indicar-hi que es poden deixar anar les targetes a dins del llistat. Finalment s'ha de fer l'event de deixar anar que dispararà la funció *drop()* declarada abans i que mourà les targetes de posició.

```
<div cdkDropList class="list"
(cdkDropListDropped)="drop($event)">
  <mat-card *ngFor="let card of cards" class="box" cdkDrag>
      <mat-card-header>
        <mat-card-title>{{card.title}}</mat-card-title>
    </mat-card-header>
    <mat-card-content>
        <iframe mat-card-image width="560" height="315"
[src]='sanitizer.bypassSecurityTrustResourceUrl(card.videoLin
k)' frameborder="0" allow="autoplay; encrypted-media"
allowfullscreen></iframe>
    </mat-card-content>
  </mat-card>
</div>
```

4.4.6 Any de copyright al peu de la web

La pàgina te un peu, on es mostren alguns elements simbòlics com l'avís legal, la política de privadesa i el copyright de Classpip amb la data d'inici i de final. Per tal de no haver d'estar canviant l'any cada cop que canviï aquest, que probablement causaria que es deixés de canviar i aquest quedi obsolet, s'ha fet una variable amb l'any actual que es mostra com a data final del copyright.

```
actualYear: number = (new Date()).getFullYear();
```

Que a la plantilla html es representa de la següent manera:

```
<p>Classpip &copy; 2016 - {{ actualYear }}</p>
```

4.4.7 Enrutament

Per a que la pàgina funcioni, ha de tenir un component que enruti les urls del navegador. Per aquesta raó es crea l'arxiu d'enrutament, amb la següent comanda de cli:

```
ng generate module app-routing --flat --module=app
```

On *--flat* col·loca l'arxiu app.routing.ts a la carpeta src/app en comptes de a la seva pròpia carpeta, i *--module=app* registra les importacions al fitxer *app.module.ts*. D'aquesta manera el fitxer es crea correctament, i amb les modificacions que s'han de realitzar als altres arxius fetes.

Les rutes disponibles a la pagina web s'han de declarar en l'arxiu *app.routing.ts*, per això s'ha de declarar la importació del component que aportarà el contingut a la pàgina i després declarar la seva ruta.

Si per exemple, es vol declarar una ruta per a la pàgina d'estudiants, es fa de la següent manera:

```
import { StudentsComponent } from
'./pages/students/students.component';

    const routes: Routes = [
        {path:'user/students', component: StudentsComponent},
    ];
```

Ara, en anar a la ruta https://classpip-onboarding.herokuapp.com/user/students es veurà el contingut del component students.component.

Aquesta pagina web es susceptible d'escriure les seves url directament al navegador, el que pot fer que hi hagi algun error a l'hora d'escriure la url i no es mostri cap contingut. Es per això que s'ha creat una pagina d'error 404 amb la seva corresponent ruta. Per a que la pàgina de 404 es mostri, s'ha hagut de crear una redirecció per a qualsevol ruta que no existeixi en aquest arxiu. L'ultima objecte de l'array de rutes es el següent:

```
{ path: '**', redirectTo: '/404' }
```

Fet això, s'assegura que si un usuari, per la raó que sigui acaba en una ruta que no existeix, tingui una indicació visual del seu error.

4.4.8 Desplegament de la web

Per a que la web d'onboarding estigui disponible públicament i sigui accessible per a qualsevol estudiant, s'ha publicat utilitzant un servidor d'Heroku en el seu pla gratuït.

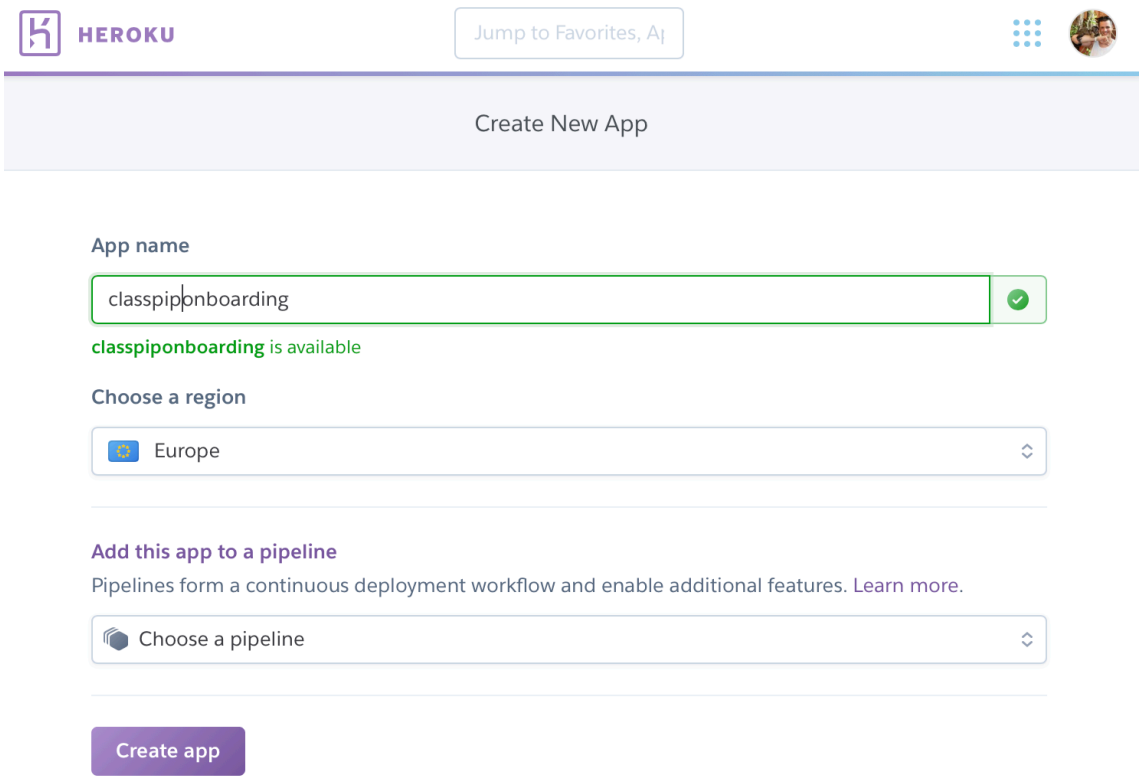
El pla gratuït te certes limitacions, però es més que suficient per a la visibilitat que es requereix en aquests moments, i es ampliable en cas de tenir necessitats superiors.

Per a poder tenir disponible a internet la web, es requereixen un passos previs, que tot seguit es detallaran.

El primer pas es crear l'aplicació web Classpip-onboarding, el seu repositori de Github i fer una sincronització del codi local amb el repositori de Github.

En aquest punt es pot veure la web localment amb la comanda 'ng serve' a la direcció *'http://localhost:4200'*. Però si es puja a un servidor web no funcionarà.

Un cop creat el repositori del projecte a Github, i havent pujat el codi a aquest, s'ha de crear una nova Web App a Heroku que allotjarà la pàgina web (Il·lustració 15). La web app s'ha creat en la regió europea per la proximitat geogràfica dels seus servidors.



Il·lustració 15: Creació de nou projecte a Heroku

Ara al menú *"deploy"*, a sota de *"Deployment method"*, es selecciona el mètode de desplegament per Github, i la web d'Heroku demanarà accés a la compte de Github que es vol relacionar (Il·lustració 16).

Tal i com es veu a la il·lustració 16, cal seleccionar la branca que es vol desplegar tant a desplegament automàtic, com a desplegament manual. En aquest cas es la branca master del repositori de Github.

Ja esta configurat Heroku per a desplegar a Heroku automàticament la app d'Angular quan es faci un push a master en Github.

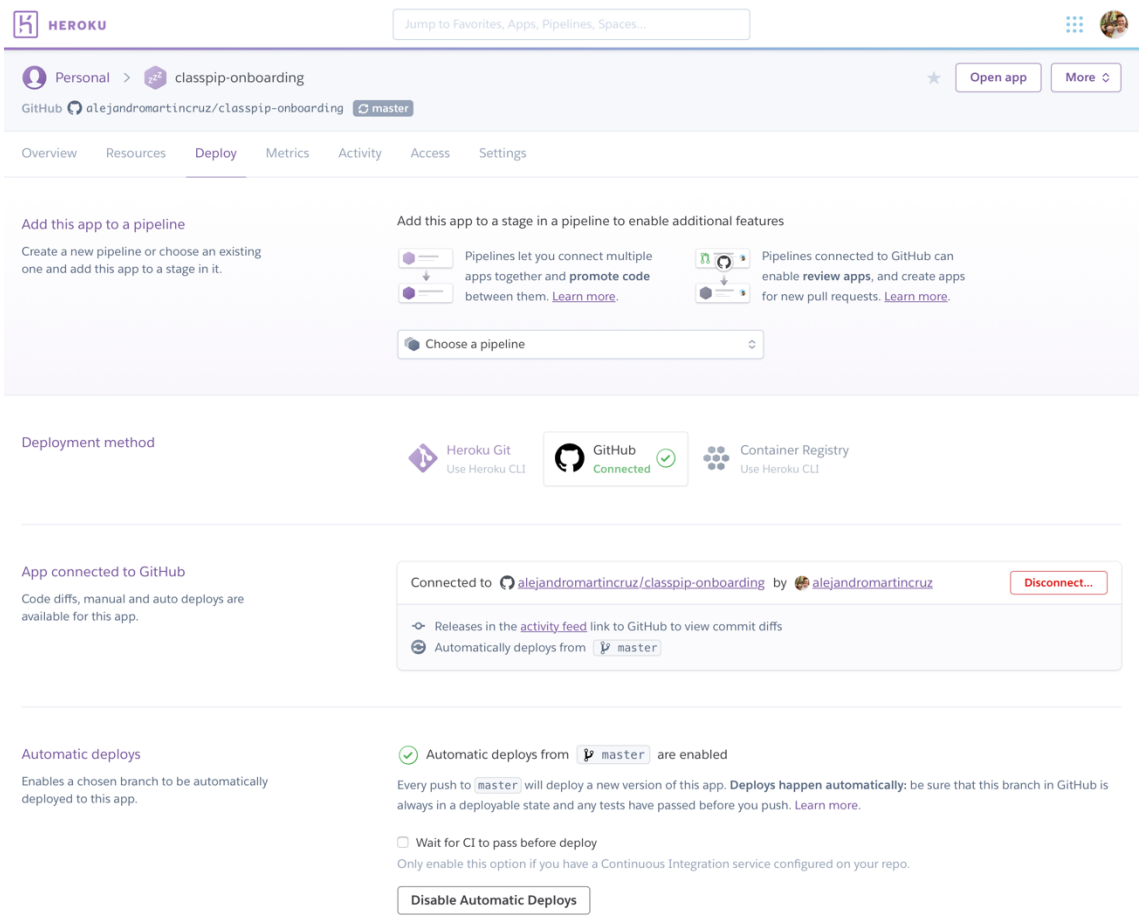
El pas final es configurar l'app d'Angular per a que funcioni el seu desplegament a Heroku.

Per això, es comença amb l'arxiu package.json on s'ha de copiar de *"devDependencies"* a *"dependències"* les línies:

```
"@angular/cli": "^7.2.2",
"@angular/compiler-cli": "^7.2.1",
"typescript": "~3.2.4",
```

Ara a sota dels scripts s'ha d'afegir una comanda de postinstall:

```
"postinstall": "ng build -prod"
```



Il·lustració 16: Configuració per a desplegament automàtic mitjançant el repositori de Github

També s'han d'afegir els motors de node i npm que utilitzarà Heroku, per aquesta raó s'han d'afegir també les següents línies:

```
"engines": {
    "node": "8.9.4",
    "npm": "6.4.1"
},
```

Ara es fa la instal·lació d'*Enhanced Resolve* i *express* amb les següents comandes:

```
npm install enhanced-resolve@4.1.0 --save-dev
npm install express path --save
```

Es procedeix a escriure un fitxer server.js a l'arrel de l'aplicació on s'utilitza el servidor express per a servir els fitxers de la web des de el directori *"dist"*. El codi del fitxer es el següent:

```
const express = require('express');
const path = require('path');

const app = express();

app.use(express.static('./dist/classpip-onboarding'));

app.get('/*', function(req,res) {

res.sendFile(path.join(__dirname,'/dist/classpip-
onboarding/index.html'));
});

app.listen(process.env.PORT || 8080);
```

Per acabar de configurar l'aplicació d'Angular per a que funcioni amb el servidor d'Heroku falta l'últim pas, que es canviar la comanda d'inici del package.json, per a que al iniciar l'aplicació utilitzi "node server.js" en comptes de "*ng serve"*. Per aconseguir això, s'ha de canviar al *"start"* dels "scripts".

En aquest moments, si es fa un *commit* a master d'aquests canvis i es puja a Github, els canvis es traslladaran a Heroku, es compilaran, i després s'aixecarà la web, disponible a internet per a tothom.

Un cop descrita la nova pàgina web de projecció externa i material d'onboarding, cal continuar descrivint els canvis realitzats al codi del dashboard per a realitzar millores. On a més hi ha una explicació del redisseny i remodelació de la pàgina *"home"* del dashboard.

CAPÍTOL 5. MILLORES DE LA WEB APP

Arribats a aquest punt, ja es disposa d'un concepte molt ampli del projecte, s'ha explicat la seva estructura, les noves funcionalitats desenvolupades i la adició i desenvolupament d'una nova web que complementarà al projecte. Però encara queden per explicar les modificacions realitzades per millorar el codi ja existent i el redisseny i remodelació que s'ha realitzat de la pàgina *"home"* del dashboard per aprofitar les noves funcionalitats de rangs, nivells i Rewards.

De cara a millorar l'experiència d'usuari, no cal només que la programació sigui correcte i funcioni bé. A més, el disseny ha de ser 'usable', es a dir, intuïtiu i senzill d'utilitzar per a qualsevol usuari. Aquest punt a vegades només comportarà canvis d'estils o de localització d'elements de la web, i unes altres vegades també comportarà cert nivell de programació per canviar algunes funcionalitats. Encara que en alguns punts, el que es realitza son millores en el codi que no s'apreciaran de cap manera en el frontal, però que faran que el codi funcioni igual o millor i sigui més sostenible i entenedor per als desenvolupadors.

Els canvis realitzats, que s'expliquen detalladament més endavant, son els següents:

- Utilització de filtres d'Strongloop a diversos llocs del projecte
- Reestructuració de la home del dashboard i creació de funcionalitats d'administrador
- Canviar l'estructura dels fitxers sass a una més fàcil d'entendre i mantenir
- Amagar el links de navegació als usuaris no loginats
- Canvi de posició del selector d'idioma a la barra de navegació
- Solució del warning de Hammer.js
- Afegir el codi repetit de options a UtilsService i substituir-ho a la resta de serveis
- Correcció d'estils i mantenibilitat del codi
- Afegir botó de tornar

5.1 Filtres de Strongloop

A la part de serveis del projecte s'utilitza Strongloop. Aquest framework per crear APIs amb Node.js aporta una eina molt potent que fins ara no s'ha utilitzat. Es tracta dels filtres, que permeten fer peticions de dades amb unes certes condicions, i les seves respostes faciliten molt el posterior tractament de dades.

Existeixen diversos tipus de filtres que es poden aplicar a Strongloop, que a més, es poden combinar entre ells. Els possibles filtres a utilitzar son:

- **Fields:** aquest filtre permet demanar a l'API només els camps que es necessiten d'un objecte. Així, per exemple, es podria demanar només el nom, el cognom i l'id de l'avatar d'un o més estudiants i no rebre cap de les altres dades guardades per a aquests. Per a fer la petició cal agregar els paràmetres del filtre al final de la url de l'API. Un exemple de les opcions es el següent:

  ```
  ?filter[fields][name]=true&filter[fields][surname]=true&filte
  r[fields][avatarId]=true
  ```

- **Include:** Aquest tipus de filtre es molt útil per a la realització del projecte. De fet, s'ha utilitzat en la funcionalitat de Rewards. Include permet incloure objectes que pertanyen a l'objecte que s'està demanant. Per exemple, fins ara, quan es volia un estudiant i el seu avatar, primer es feia una consulta a l'api demanant l'estudiant, i després es feia una altre consulta utilitzant l'avatarId de l'estudiant, per obtenir el seu avatar. Gracies a l'include, es pot fer que quan es demana l'estudiant ja vingui amb l'avatar inclòs, estalviant la segona consulta. Si per exemple es vol fer la consulta de l'estudiant amb l'avatar, cal incloure els següents paràmetres ala url de la consulta:

  ```
  ?filter[include]=avatar
  ```

- **Limit:** aquest filtre permet limitar el nombre d'objectes que s'obtindran a la resposta. Així, per exemple, si es demanen els estudiants, amb el límit es pot limitar la resposta a un nombre concret d'estudiants en comptes del llistat complert. Si es vol limitar el llistat d'estudiants a 5, es faria incloent els següents paràmetres al final de la url:

  ```
  ?filter[limit]=5
  ```

- **Order:** aquest filtre permet obtenir els elements de la resposta ordenats per algun dels seus camps. Ha sigut de molta utilitat en el mòdul de rangs i en el de Rewards, ja que permet obtenir el llistat de rangs ordenat pel nombre de punts necessari per assolir el rang, i d'aquesta forma fa innecessari processar la resposta per a realitzar aquesta ordenació. Per a fer aquesta ordenació en els rangs, cal afegir els següents paràmetres:

  ```
  ?filter[order]=puntosRango%20ASC
  ```

- **Skip:** aquest filtre permet ometre els primers n resultats de la resposta. Per si sol podria no semblar molt útil, però utilitzat conjuntament amb el filtre de límit pot servir per a fer paginació en els llistats. Per exemple, si es limiten els resultats a 10, amb el filtre de **limit**, i s'ometen els primers 10 resultats amb el filtre d'**skip** es com fer una paginació a la segona pàgina. Si s'ometen els primers 20 resultats, es com estar a la tercera pàgina, etc.

- **Where:** aquest filtre també es molt útil, ja que permet obtenir resultats on un o mes camps compleixen un requisit. Algunes de les opcions que permet son: 'major que' (gt), 'menor que' (lt), 'entre' (between), 'com' (like), etc. Per exemple, es poden llistar tots els estudiants que estan assignats a l'escola amb id = 1, o el que es el mateix que el seu schoolId es igual a 1. En aquest cas, el que s'ha d'afegir a la url es:

  ```
  ?filter[where][schoolId]=1
  ```

La utilització d'aquests filtres pot simplificar molt el codi per a tractar les dades que s'utilitza després d'obtenir la resposta de l'API.

Tot i això, Strongloop no te desenvolupament des de ja fa dos anys, i n'hi ha algunes limitacions que no es van solucionar abans d'aturar el desenvolupament. La que més ha afectat es no poder fer el filtre include a un segon nivell de dades. Es a dir, el filtre include permet demanar un llistat de professors o estudiants i que cada un d'ells vingui amb el seu avatar corresponent inclòs. També es pot demanar una escola que inclogui els seus professors i estudiants. Però no es pot demanar una escola que inclogui els professors i estudiants, i que, a la vegada aquests incloguin els seus avatars.

5.2 Reestructuració de la home

La pàgina d'inici del dashboard estava molt infrautilitzada, i no mostrava gens d'informació útil als usuaris. Gairebé era una pàgina de la que es podia prescindir sense cap afectació al projecte.

Amb el motiu de millorar la funcionalitat de la home, s'ha tornat a redissenyar i s'ha refet aquesta pàgina. Les reformes principals en aquest punt son:

- Oferir un perfil d'usuari vistós i atractiu als estudiants que els mostri la informació dels seus avenços de forma clara, i a la vegada els motivi a aconseguir noves fites.

- Mostrar al professor un resum dels estudiants i els seus progressos, a més permet fer diverses ordenacions de les dades segons els interessos del professor i que d'aquesta manera li sigui molt mes fàcil extreure estadístiques dels estudiants.

- S'ha creat una vista per a l'usuari administrador. Ja que fins ara, les dades amb les que es treballava eren dades prefabricades en un arxiu de l'aplicació de Servei, i no hi havia manera de crear escoles, professors, cursos, etc. A partir d'ara, l'administrador podrà tenir control de tot el contingut.

Pàgina home per a estudiants:

La pàgina home per a estudiants, ara mostra molta informació a primera vista, d'una forma vistosa i entenedora (Il·lustració 19).

Començant per la part superior, l'estudiant veu el seu avatar i el seu nom complert. Just a la dreta l'estudiant veu una barra de progrés amb els punts que necessita per a arribar al següent nivell. Aquesta barra que mostra el **percentatge** aconseguit del següent nivell, s'anirà emplenant segons l'estudiant vagi aconseguint nous punts.

Just a sota d'aquesta informació, hi ha tres grans requadres. El primer mostra el rang que ha aconseguit l'estudiant amb la imatge que identifica aquest rang i el títol que li pertoca. Al segon requadre, l'estudiant veu molt destacat el nivell actual que ha aconseguit amb una font gran i de color vermella, i a sota els punts actuals que ha aconseguit. Al tercer i últim requadre, l'estudiant pot veure les icones dels objectes punts amb el numero de punts de cadascuna que ha aconseguit.

Tot seguit, esta el ranking d'estudiants. En aquesta **classificació** es mostren els tres millors estudiants de l'escola, amb la posició que ocupen a aquest ranking sobre la seva fotografia, i a més el seu nivell actual, els punts totals que han obtingut i el seu rang. Aquesta classificació te una paginació que permet avançar en el llistat, de manera que l'estudiant pot anar paginant veient les següents posicions fins a veure la seva pròpia posició. Quan l'estudiant vegi la targeta corresponent a la seva posició la podrà identificar amb el tag "la meva posició" que es mostra sobre la targeta, i que a la il·lustració 19 es veu sobre la primera targeta.

Per últim, podrà veure informació relativa a la seva escola, així com enllaços cap a les xarxes socials d'aquesta.

Pàgina home per a professors:

En aquesta pàgina (Il·lustració 20) el canvi principal es la **classificació** d'estudiants. Aquest ranking d'estudiants presenta una sèrie d'informació molt útil per al professor. En aquest cas es tracta d'un llistat complert d'estudiants, on el professor pot veure informació relativa als progressos dels estudiants en una taula amb scroll horitzontal.

D'esquerra a dreta, el que veu el professor es l'avatar de l'estudiant, el seu nom, el seu cognom, la quantitat de punts que te l'estudiant, el seu nivell, el seu rang,

i finalment una columna per a cada tipus de punt, i la quantitat de punts que ha obtingut cada estudiant de cada un d'aquests punts.

De cara al professor el més interesant d'aquesta taula es poder organitzar els resultats per la columna que l'interessi. Doncs el professor podrà mostrar els estudiants per ordre creixent o decreixent de cada una de les columnes de la taula, excepte la dels avatars. Per exemple, el professor pot decidir ordenar la taula pel cognom dels estudiants per ordre creixent, o també pot ordenar els alumnes pels punts de retràs en ordre decreixent per veure quins son els alumnes que arriben tard més cops.

A més el llistat esta paginat per evitar obtenir un llistat massa llarg que no es fàcil de llegir. En la paginació el professor també pot triar quant alumnes vol veure alhora 5, 10, 25 o 100, segons quina quantitat d'alumnes simultània li sigui més còmode.

Pàgina home per a administradors:

No hi existia una pàgina per a l'administrador abans, per el que aquesta s'ha hagut de dissenyar i generar partint des de zero. Aquesta pàgina d'inici (Il·lustració 21) consisteix en les dades d'identificació de l'usuari, i tot seguit un llistat de les escoles creades a Classpip.

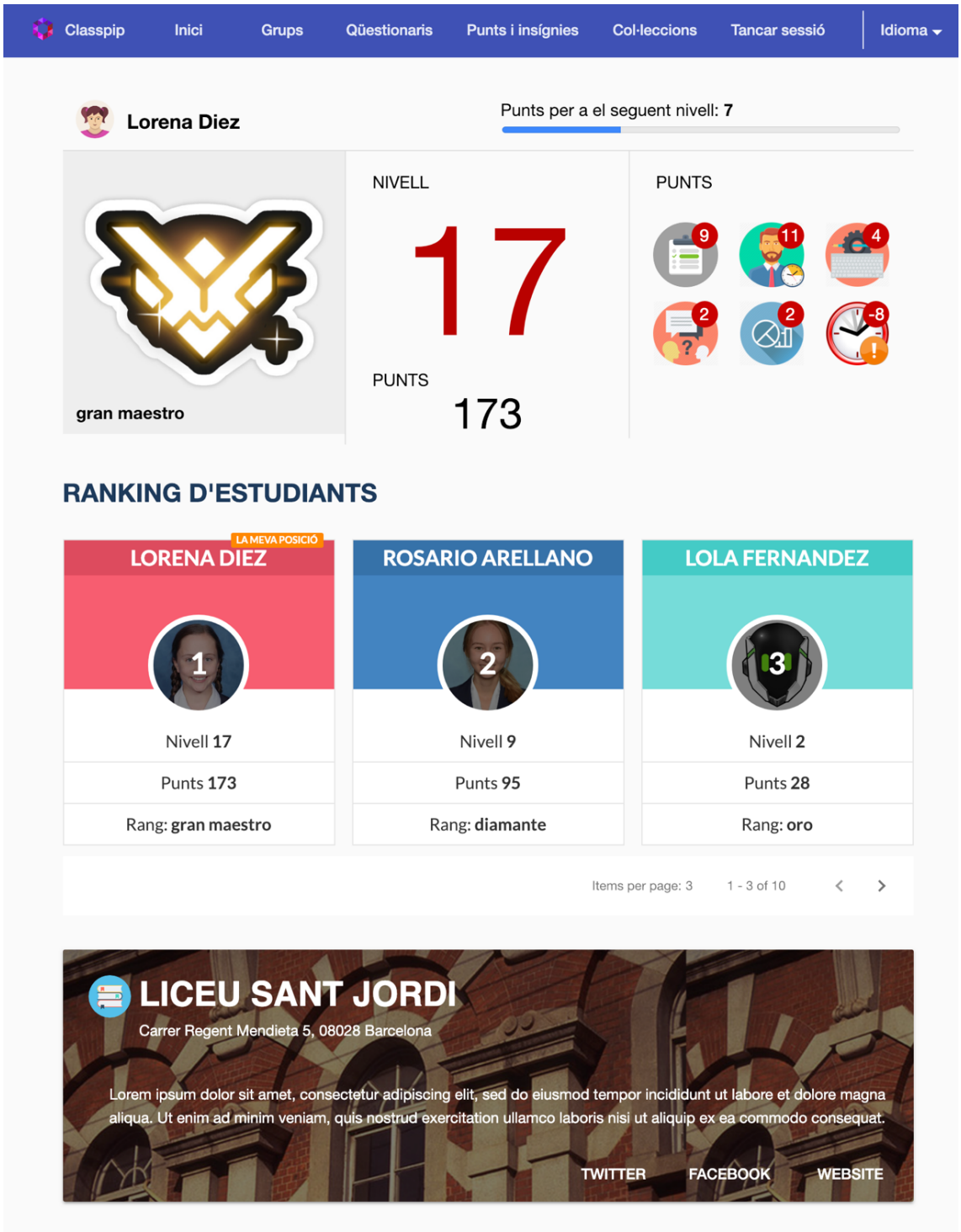
En el llistat es pot veure un botó gran per a crear una nova escola, que en clicar-ho farà scroll fins a col·locar el formulari de nova escola a la vista de l'administrador.

A cada un dels elements del llistat, es pot veure informació bàsica sobre l'escola, i dos botons. El primer es el botó d'editar, en polsar-lo, la pàgina farà scroll fins al formulari, canviant el seu títol per un que indiqui que s'està editant l'escola escollida i mostrarà als seus camps les dades de l'escola que es vol editar. I finalment, el botó d'eliminar, que en polsar-lo mostrarà primer una finestra de confirmació, i en acceptar-la procedirà a esborrar l'escola.

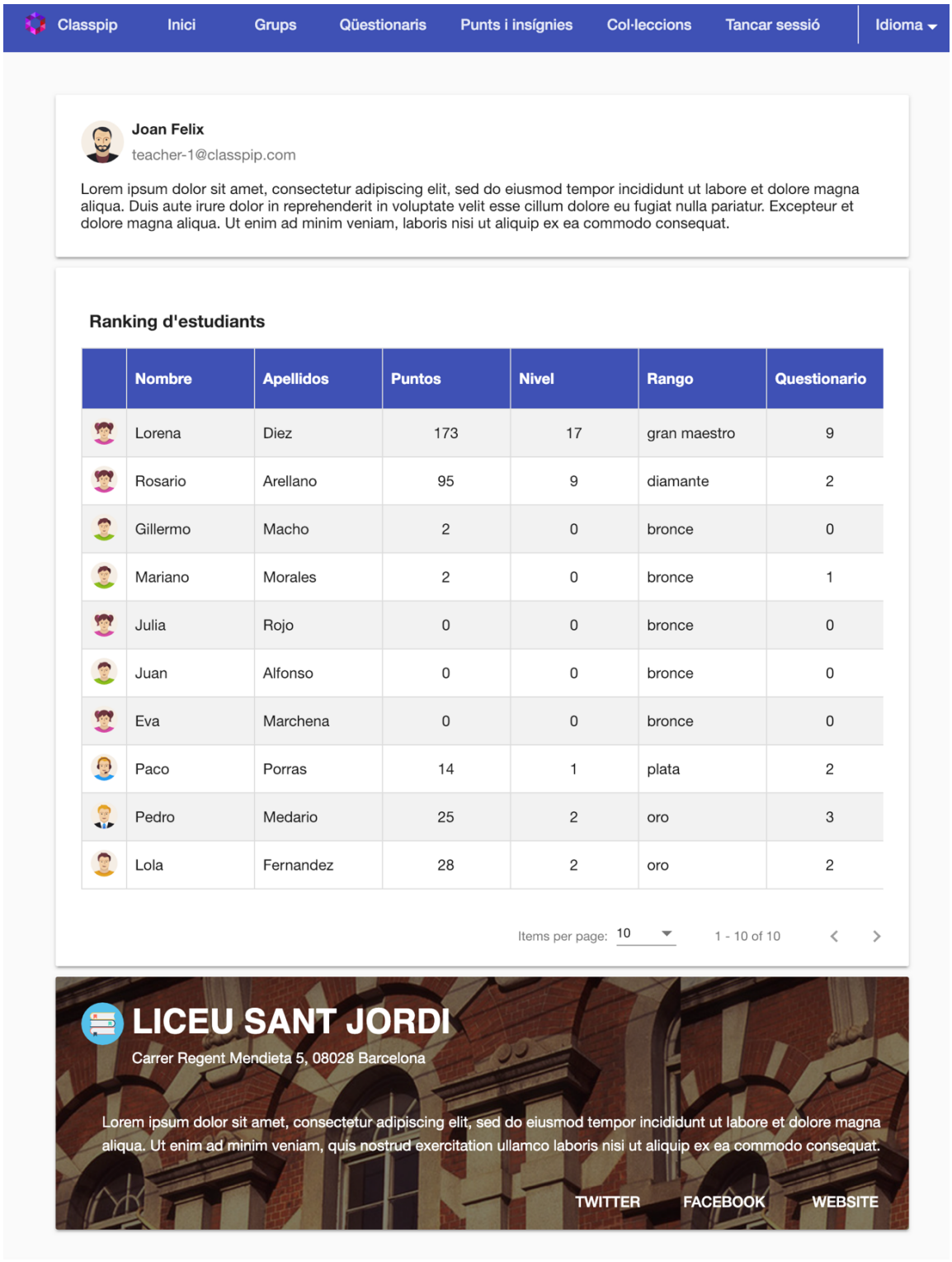
Cada una de les escoles llistades, es un element clickable, tal i com indica el cursor en el seu format "pointer". En fer click a l'escola, l'administrador anirà a una pàgina de gestió d'elements de l'escola. I en aquest lloc podrà generar professors, estudiants, rangs, cursos, assignatures, etc.

El funcionament de cada un d'aquests llistats (Il·lustració 22) es exactament el mateix que el funcionament del llistat d'escoles, amb l'excepció de que l'element sigui clickable, ja que a aquests llistats aquests no son clickables. D'aquesta manera, hi ha una uniformitat a l'hora de gestionar aquests elements i a l'usuari li serà molt fàcil aprendre a utilitzar-los.

A nivell de codi també hi ha hagut molts canvis en la reestructuració de la home. El principal i més visible es la reducció de codi a la plantilla html de home de 31 línies a només 5.



Il·lustració 17: reestructuració de la pàgina d'inici per als estudiants



Il·lustració 18: reestructuració de la pàgina d'inici per als professors

Això ha sigut possible gràcies a la divisió del seu contingut en components que després s'han inclòs dins d'aquest, el que fa el codi molt més llegible i reutilitzable. Encara que, en alguns casos, ha suposat un repte transferir dades entre els components pares i fills en les dues direccions.

El codi de la plantilla de la home ha quedat de la següent manera:

```
<div class="home-content">

  <app-admin-home *ngIf="isAdminProfile && profile"
[profile]="profile"></app-admin-home>

  <app-teacher-home *ngIf="isTeacherProfile && (profile && school
&& allPoints)" [profile]="profile" [school]="school"
[allPoints]="allPoints"></app-teacher-home>

  <app-student-home *ngIf="isStudentProfile && (profile && reward
&& rank && studentPoints)" [profile]="profile" [reward]="reward"
[rank]="rank" [studentPoints]="studentPoints"
[school]="school"></app-student-home>

</div>
```

Ara hi ha un component per a cada un dels casos, en comptes de ficar tot el codi de cop en una sola plantilla.

Per a passar dades d'un element pare a un element fill només cal posar-lo a la plantilla quan es crida el component fill de la següent manera: *[profile]="profile"* i al l'arxiu de codi typescript del component fill s'ha d'importar Input i declarar la variable entrant com a: *@Input() profile: Profile;*

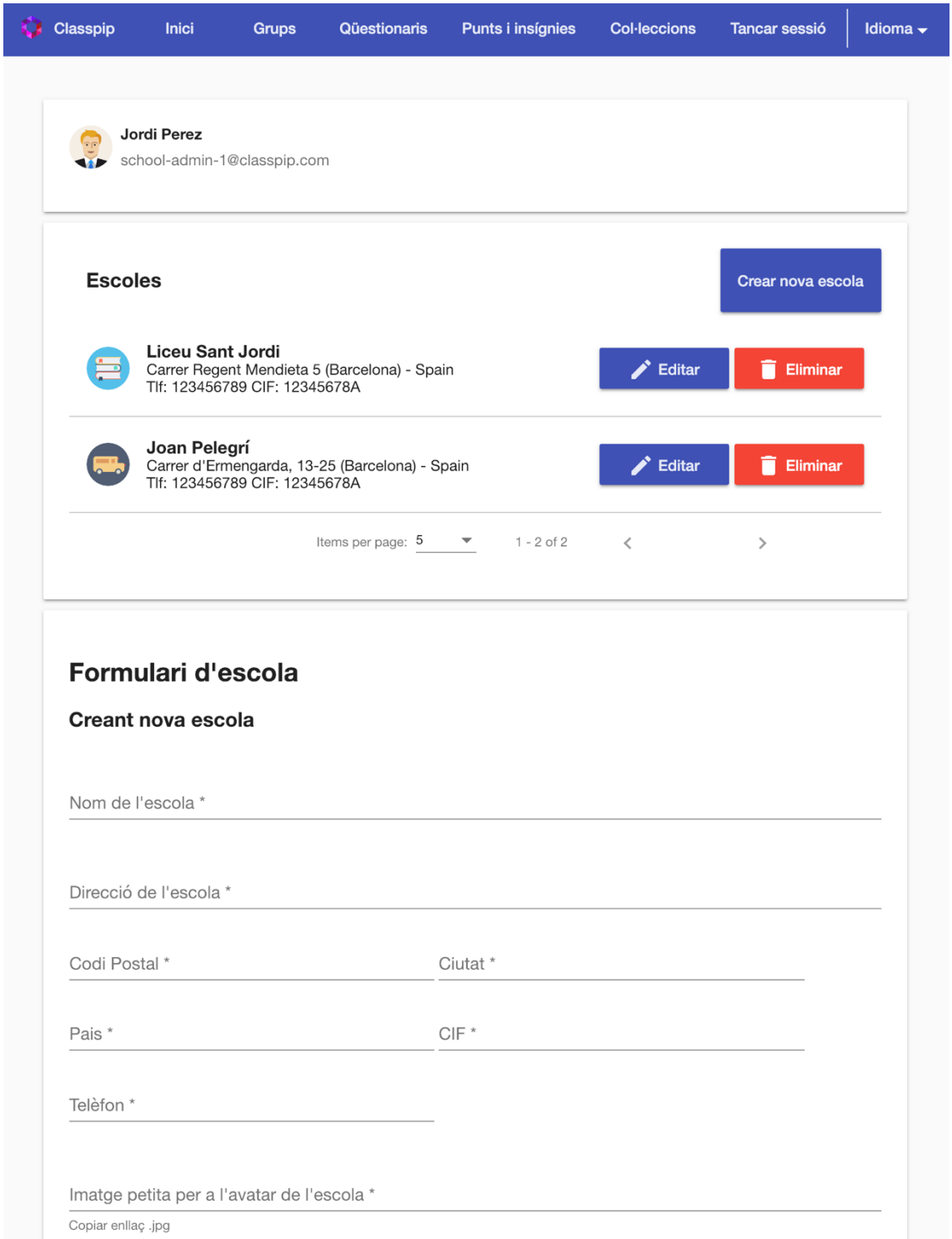
Els formularis de creació i edició d'escola, i dels altres elements a gestionar, també s'han realitzat en components per separat, i en aquest cas la comunicació entre el component pare i el component fill es bidireccional. Es a dir, que es passa informació cap als fills de la manera indicada abans, però també es passa informació des de els components fills cap als pares.

Com a exemple s'utilitzarà el formulari d'escola, que es declara al component pare, en aquest cas *admin-home*, com a *app-school-form* de la següent manera:

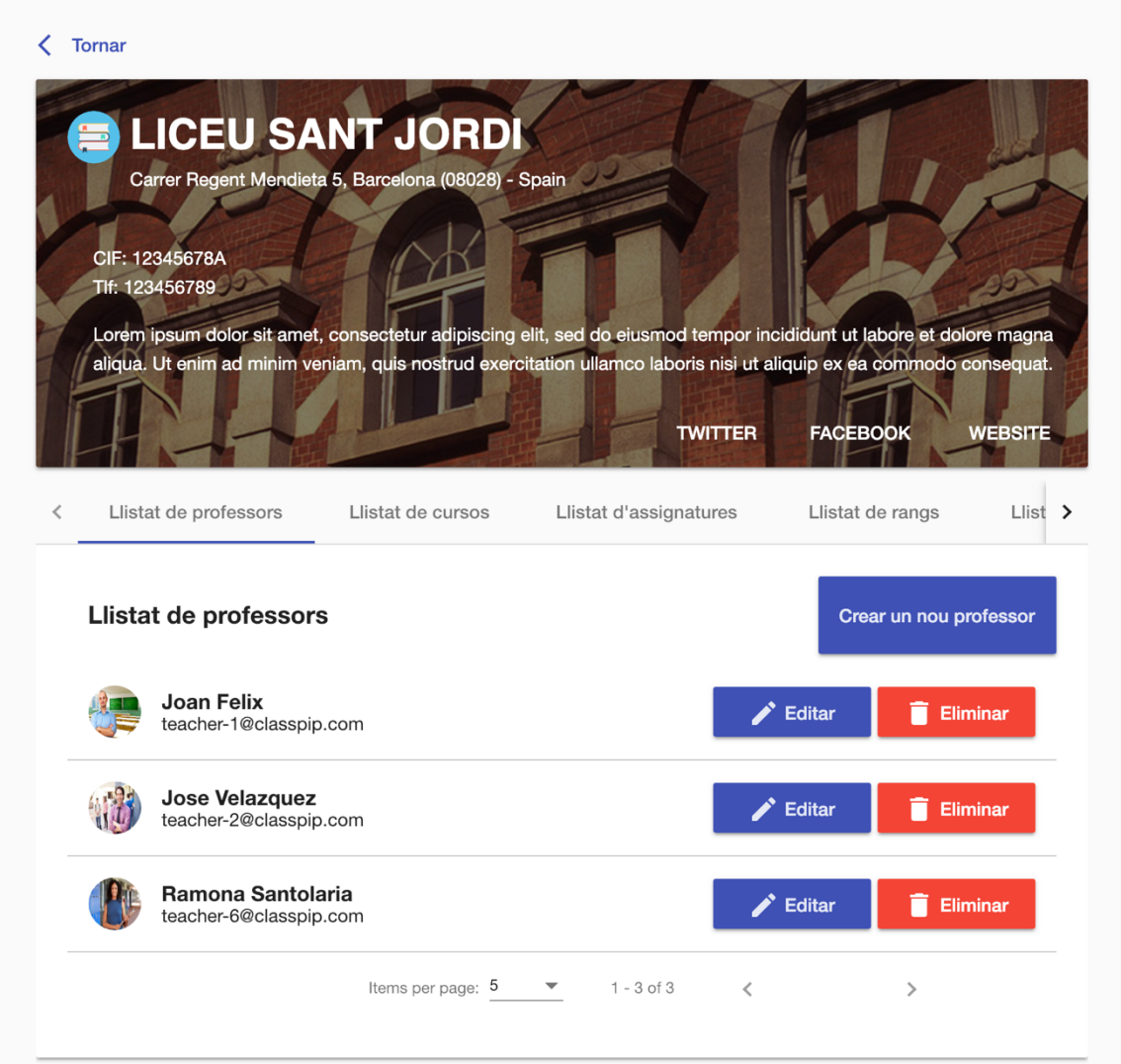
```
<app-school-form [editSchool]="editSchool"
(newSchoolChange)="newSchoolHandler($event)"></app-school-form>
```

Aquí es veu com el component pare passa la variable *editschool* cap al formulari, i mitjançant un observer d'un event es passa informació sobre *newSchool* cap al component pare.

En el component fill, el formulari, s'ha d'importar Output. Amb l'output, s'ha de declarar el event que s'emetrà de la següent manera: *@Output() newSchoolChange = new EventEmitter<School>();* finalment, quan es necessiti, es llençarà l'event de la següent manera: *this.newSchoolChange.emit(school);* En aquest cas, el event es llençarà quan es generi exitosament una nova escola, i contindrà l'objecte School que retorna l'API.



Il·lustració 19: vista de la pàgina d'inici per als administradors



Il·lustració 20: pagina de gestió dels elements d'una escola

Ara, al pare s'ha de declarar la funció que s'executarà quan el event es dispari en el component fill.

```
newSchoolHandler(newSchoolGenerated) {
    this.schools.push(newSchoolGenerated);
    this.pageEvent.length += 1;
}
```

Amb aquesta configuració, després de crear exitosament una nova escola, es dispara un event *newSchoolChange* amb l'objecte escola que s'acaba de crear. En aquest moment l'observer del component pare executa la funció *newSchoolHandler* amb la nova escola com a paràmetre.

N'hi ha un altre cas peculiar generat pel fet d'utilitzar el formulari en un component separat, i es que quan es pressiona sobre el botó editar, i s'assigna

a *editSchool* l'objecte escola que es vol editar, el component fill no s'adona del canvi en la variable *editSchool*.
Per aquesta raó s'ha hagut d'implementar *OnChanges* al formulari. *OnChanges* es un dels lifecycle hooks d'angular, que realitzen accions en determinats moments. Aquest' realitza accions quan detecta canvis en els paràmetres indicats. El paràmetre que s'indica en aquest cas, es *SimpleChanges*, que detecta quan una propietat ha canviat. Durant aquest procés es comprova si la propietat que ha canviat es *editSchool*, i en cas afirmatiu s'assigna el seu valor a *newSchool* per a que empleni el formulari amb les dades de l'escola.

```
ngOnChanges(changes: SimpleChanges) {

   for (const propName in changes) {
     const change = changes[propName];

     if (propName === 'editSchool') {
       if (this.editSchool !== undefined) {
         this.newSchool = this.editSchool;
       }
     }
   }
}
```

5.3 Estructura sass dashboard

En el dashboard, els arxius sass estan posats a diversos llocs i carpetes, de forma que fer modificacions de css al projecte no son intuïtius, ni fàcils. I no existeix una clara separació de les diverses eines que ofereix sass.

Originalment existeix un arxiu main.scss a la carpeta src, on es declaren les variables, i es fan diverses importacions.

Un arxiu _general.scss dins de la carpeta src/styles on hi ha alguns estils, però que s'importa en el arxiu main.scss abans del tema, per el que tots els canvis que es facin en aquest arxiu i que després tinguin algun estil al tema referent al mateix element, el resultat serà que l'element tindrà l'estil del tema, i no l'introduït a l'arxiu _general.scss.

Hi existeix un altre arxiu a src anomenat _app-theme.scss on s'ha implementat tot l'estil de la pàgina dins d'un mixin.

Els canvis a realitzar en aquest apartat, es poden veure a **la branca 'sass_folder'**, i son crear una carpeta sass dins de la carpeta src. En aquesta carpeta es crea un arxiu anomenat styles.scss, un arxiu anomenat _variables.scss i un arxiu anomenat _mixins.scss.

Ara cal anar a l'arxiu .angular-cli.json i canviar l'array styles de main.scss, l'arxiu utilitzat fins ara, per sass/styles.scss. Amb aquest canvi se l'indica a l'aplicació que ja no utilitzi més el main.scss i en el seu lloc utilitzi el nou arxiu que s'ha creat. Com que l'arxiu main.scss ja no s'utilitza i ja no es necessari s'ha d'esborrar per evitar confusions.

A partir d'ara totes les variables es declararan en l'arxiu de _variables.scss, lloc on es podran localitzar i modificar molt fàcilment. Tanmateix, a l'arxiu de _mixins.scss es declararan totes les directives mixin que després es vulguin utilitzar.

A l'arxiu styles.scss s'importa el tema d'Angular Material, i els arxius de variables i mixins. Es crea l'objecte $theme amb les variables declarades a l'arxiu de variables i després s'inclouen els estils de tema utilitzant l'objecte $theme. A partir de les següents línies es pot introduir tot el codi css en format sass que es desitgi.

Finalment s'afegeixen dos arxius més a la carpeta de sass, un per els media quèries de tablet i un altre per els media quèries de desktop. El codi d'estils s'haurà d'escriure pensant en "Mobile first", això significa que el codi d'estils que s'escriurà normalment serà el de la versió mòbil, i després s'adaptarà a la versió tablet i desktop utilitzant els media quèries localitzats als arxius de tablet i desktop.

Per exemple, si es vol que un element ocupi tota l'amplada a mobile, es posarà l'estil 'width: 100%' per a aquest element a l'arxiu styles.scss a la zona on esta indicat amb els comentaris que s'ha de posar el codi css i abans de les línies on s'importen els arxius de tablet i desktop.

Si es vol que aquest mateix element tingui una amplada de mitja pantalla en tabletes, s'haurà de posar 'width: 50%' per a aquest element a l'arxiu _tablet.scss dins el media query destinat a aquesta mida.

I si finalment es vol que aquest element a desktop nomes ocupi una quarta part de l'amplada, només caldrà afegir 'width: 25%' per a aquest element a l'arxiu _desktop.scss dins el media query destinat a aquesta mida.

Amb aquesta estructura d'arxius es molt més fàcil afegir o canviar variables, crear nous mixins o localitzar els existents per poder utilitzar-los, i també crear nou codi css adaptable a Mobile, tablet i desktop.

5.4 Amagar links del menú per usuaris no loginats

Fins ara, a la web del dashboard es mostraven sempre tots els links de navegació a la capçalera de la web. Tant per a usuaris loginats, com per a usuaris no loginats.

Això comporta un problema d'usabilitat, ja que els usuaris no loginats, en accedir veuen tota una sèrie de links que poden polsar (Il·lustració 17 imatge superior), però no hi no podran accedir-hi, ja que només els usuaris loginats tenen accés. A més també es mostrà l'enllaç de tancar sessió als usuaris que no l'han iniciada, cosa que també pot aportar certa confusió.

La solució es molt senzilla, i es, amagar tots aquests enllaços als usuaris no loginats (Il·lustració 17 imatge inferior), i que només apareguin als usuaris que han iniciat sessió prèviament.

Per a fer aquest canvi, s'ha modificat el servei de login, el servei user i els elements del component navbar.

Els canvis d'aquest punt, es poden veure a la branca **'hide_navlinks'**.

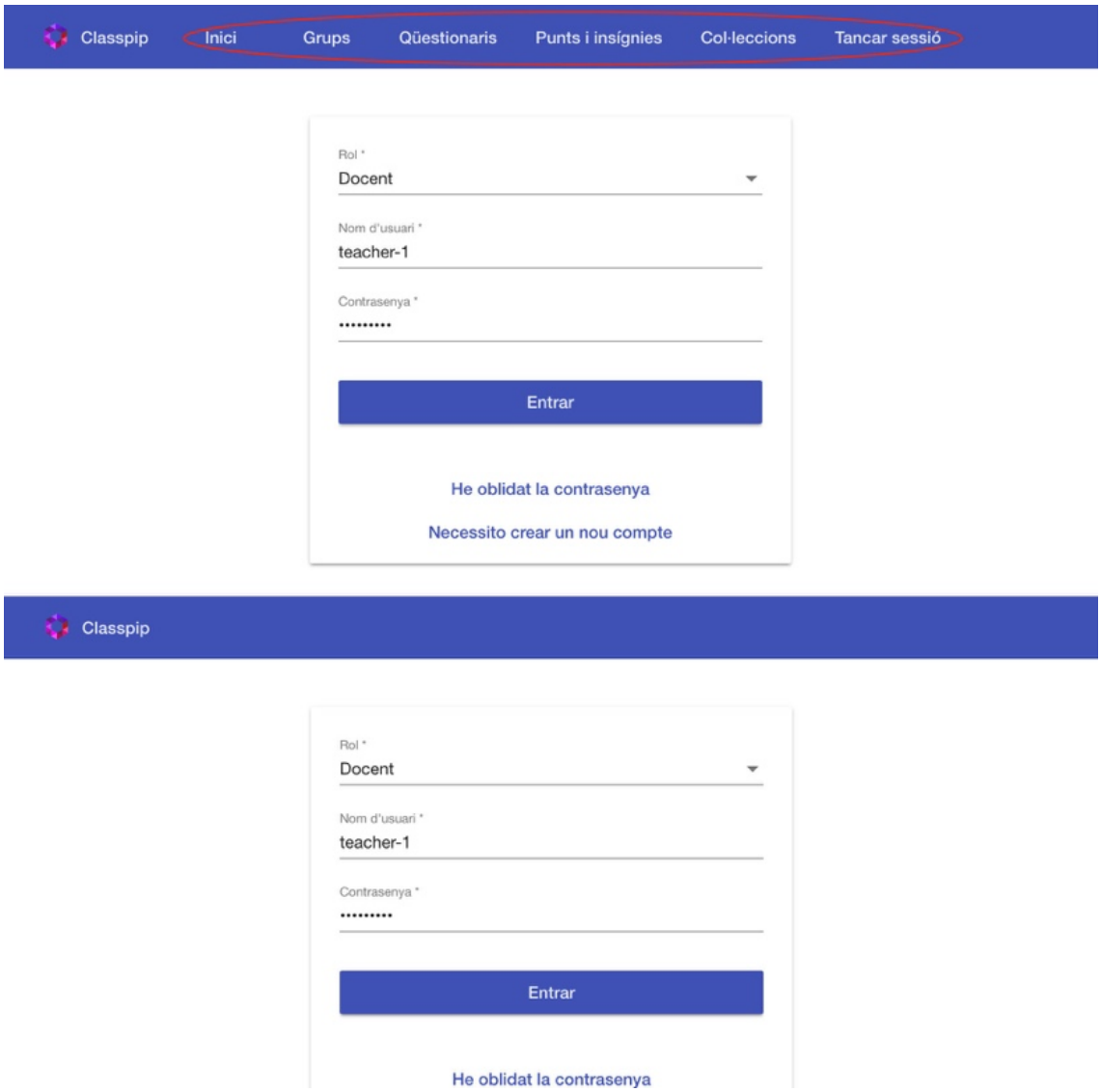
Els primers canvis s'han fet a l'arxiu app/shared/services/login.service.ts, i els canvis han sigut:

- Importació de BehaviorSubject:

```
import {BehaviorSubject} from 'rxjs/BehaviorSubject';
```

- Declaració d'un atribut públic anomenat *'loggedIn'* amb un objecte *BehaviorSubject* booleà:

```
public loggedIn = new BehaviorSubject<boolean>(false);
```



Il·lustració 21: solució als enllaços mostrats a usuaris no loginats

- Establir un *getter* que retorni l'atribut *'loggedIn'* com a observable:

```
get isLoggedIn() {
    return this.loggedIn.asObservable();
}
```

- Assignar a l'atribut *'loggedIn'* el valor 'true' al mètode de login() i el valor 'false' al mètode de logout() quan l'acció de login o logout es realitza correctament:

```
this.loggedIn.next(true);
this.loggedIn.next(false);
```

Amb l'atribut que indica si un usuari esta loginat o no amb un observable, ara resta subscriure's a aquest en el component, per a obtenir el valor quan canviï. Primer s'han de fer unes modificacions a l'arxiu typescript del component del navbar:

- S'afegeix OnInit a la importació, i també s'importa el LoginService

```
import { Component, OnInit } from '@angular/core';
import { LoginService } from '../services/login.service';
```

- S'afegeix el servei de login al constructor:

```
constructor(
    private loginService: LoginService
  ) { }
```

- S'afegeix una variable de component on s'emmagatzema el valor booleà que indica si l'usuari ha iniciat sessió o no. I es crea una funció que es subscriu als canvis de l'atribut del servei de login.

```
public isLoggedIn: boolean;

userIsLoggedIn() {
    this.loginService.isLoggedIn.subscribe(
      ((res: boolean) => {
        this.isLoggedIn = res;
      })
    );
  }
```

- S'ha d'implementar OnInit i cridar la funció de subscripció quan es dispari el callback ngOnInit.

```
export class NavBarComponent implements OnInit {

  [...]

  ngOnInit(): void {
    this.userIsLoggedIn();
  }
  [...]
}
```

- Per acabar amb els canvis del component navbar, s'ha de modificar la plantilla html afegint un condicional (*ngIf="isLoggedIn") a tots els enllaços que només s'han de mostrar quan l'usuari ha iniciat sessió.

Amb aquests canvis, ja s'aconsegueix que els links del menú de navegació no es mostrin, i es tornin visibles només quan un usuari inicia sessió. Però com que l'atribut *'loggedIn'* s'inicia com a 'false' i es canvia durant l'acció de login i logout, quan la pàgina recarrega, l'atribut s'inicia en 'false' i l'usuari no executa l'acció de login o logout, així que es queda com a 'false' el menú desapareix encara que l'usuari tingui la sessió iniciada.

Per a solucionar aquest comportament, s'ha de modificar l'arxiu app/shared/services/user.service.ts, i el canvi a realitzar es el següent:

- Si hi existeix una id d'inici de sessió, llavors l'usuari esta loginat i es força un canvi a 'true' en l'atribut 'loggedIn'. Aquesta comprovació es realitza en la funció getProfile().

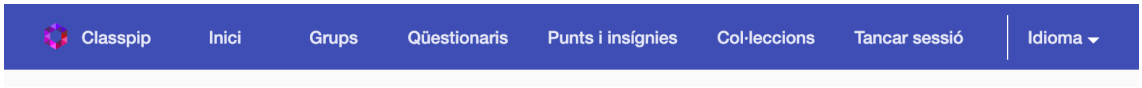
```
if (this.utilsService.currentUser.id) {
    this.loginService.loggedIn.next(true);
}
```

5.5 Canvi de posició del selector d'idioma

Existeix un selector per a seleccionar l'idioma de visualització de la pagina, aquest selector es llençava únicament des de la pagina home, i el botó per a poder seleccionar idioma no era a cap altre lloc.

L'usuari pot arribar a iniciar la navegació sense fixar-se en aquest botó de la pàgina home, amb el que si despès vol canviar d'idioma no sabrà on fer-ho.

Per aquest motiu, s'ha eliminat el botó del selector d'idioma de la pàgina home i s'ha introduït aquest en el menú de navegació (Il·lustració 18). D'aquesta manera, si un usuari vol canviar d'idioma sempre veurà on es el botó per a seleccionar l'idioma que vulgui.



Il·lustració 22: nova barra de menú amb el selector d'idioma

5.6 Warning de Hammer.js

A la consola del navegador es veia constantment un warning avisant de que faltava hammer.js.

"Could not find HammerJS. Certain Angular Material components may not work correctly."

Angular Material requereix un mòdul anomenat hamme.js per a que alguns dels seus components funcionin (mat-slide-toggle, mat-slider, matTooltip), això significa que si no s'instal·la i no s'utilitza cap dels components que requereixen hammerjs tot funcionarà bé, però a la consola sortirà constantment un warning avisant de que s'ha d'instal·lar, per contra, quan s'utilitzi alguns d'aquests components aquest no funcionarà.

Els canvis realitzats per aquest punt, al ser menors, s'han inclòs en **la branca 'sass_folder'**.

Solucionar aquest warning es tan senzill com realitzar la instal·lació de hammerjs amb npm i incloure l'import a l'arxiu main.ts

Instal·lació de hammerjs:

```
npm install --save hammerjs
```

Afegir línia a l'arxiu main.ts:

```
Import 'hammerjs';
```

5.7 Afegir codi de options a UtilsService

N'hi ha un fragment de codi que es repeteix constantment a tots els serveis:

```
const options: RequestOptions = new RequestOptions({
    headers: this.utilsService.setAuthorizationHeader(new
Headers(), this.utilsService.currentUser.id)
});
```

Segons el principi de disseny de codi DRY (Don't Repeat Yourself), això es una mala pràctica fàcilment evitable. I de fet, la solució ha estat posar aquest codi a UtilsService retornant el valor que es necessita. Un cop fet això ja es pot substituir aquestes línies recurrents de codi per només aquesta:

```
const options = this.utilsService.getOptions();
```

Aquest canvi tan senzill suposa que si al dashboard n'hi havia vint serveis, i a cada un d'aquest serveis aquest codi es repetia una mitjana de deu cops a cada un feia un total de 300 línies de codi absolutament iguals. Ara, aquestes tres línies es resumeixen en una que crida el servei d'una única línia, el que resulta en 100 línies, i suposa un estalvi de 200 línies de codi.

5.8 Correccions d'estil i mantenibilitat de codi

Tal i com s'indica a la guia d'instal·lació de l'entorn de programació, amb el Visual Studio Code es recomana la instal·lació de TSLint.

TSLint es una eina d'anàlisis del codi, que comprova la qualitat del codi typescript escrit comparant-lo amb una sèrie de regles fixades. TSLint comprova el codi per a millorar la seva llegibilitat, mantenibilitat i errors de funcionalitat.

El codi estava correcte i funcionava be, però segons les indicacions de TSLint hi havia moltes millores a realitzar en el codi. Sobretot a la part dels serveis del dashboard, on hi havia moltes coses per millorar.

Son petits canvis que no es fan notar a l'hora d'executar el programa, però que donen una uniformitat al codi que faran que futurs estudiants puguin llegir i entendre una mica més fàcilment algunes coses del codi al mantenir aquest el

mateix estil a tot el projecte, tot i haver sigut desenvolupat per moltes persones diferents.

Entre tots els canvis realitzats, els més comuns i destacats han sigut:

- La utilització de 'var' per a declarar variables. A typescript no s'utilitza mai 'var', en el seu lloc s'utilitza 'const' per a declarar una variable que sempre mantindrà el mateix valor, i 'let' per a declarar una variable que canviarà el seu valor. Així, s'han substituït tots els 'var' per 'let' o 'const'.

- La utilització de 'let' per a variables que no canvien mai. En aquest cas s'han substituït per 'const'.

- Eliminació de variables declarades que no s'utilitzaven en cap moment. A l'hora de desenvolupar pot passar que es declarin unes variables i s'utilitzin, però després de refactoritzar el codi i millorar-lo aquestes deixin de ser necessàries, i a vegades aquestes no s'esborren i queden al codi sense cap utilitat.

- Finalització de les línies de codi amb punt i coma. Un dels avantatges de JavaScript es que, normalment funciona perfectament sense aplicar els punt i coma al final de cada línia de codi, encara que si que hi pot haver alguna situació on la falta del punt i coma pot provocar alguna diferencia en el codi. I es per aquesta raó, i per llegibilitat, que s'han de finalitzar sempre les línies de codi amb punt i coma. S'han afegit els punts i coma de totes les línies que no ho tenien.

- Eliminació de parts del codi que estaven comentades i no s'utilitzaven. A vegades es deixen com a referencia mentre es fa un canvi, però un cop finalitzat s'haurien d'esborrar per facilitar la lectura del codi, a més el codi canviat o eliminat quedarà reflectit al repositori de git.

5.9 Botó tornar

En arribar a algunes pàgines, no hi havia forma de tornar enrere, fent que l'usuari perdés la referencia d'on esta i obligant-lo a clicar sobre els enllaços principals tornant a iniciar el flux de navegació.

Aquest problema té una molt senzilla solució, i es utilitzar un botó de 'tornar' que porti a la pàgina anterior, fent que no es perdi la referencia de navegació.

La implementació d'aquest botó també es molt senzilla, primer s'ha de fer la següent importació:

```
import { Location } from '@angular/common';
```

Després es declara al constructor:

```
constructor( public location: Location ) {  }
```

Ara es construeix una funció molt simple:

```
goBack() {
    this.location.back();
  }
```

Finalment es crea un botó que en ser clickat cridi la funció de tornar enrere:

```
<button mat-icon-button (click)="goBack();" color="primary"
class="go-back-btn">
      <mat-icon>arrow_back_ios</mat-icon>
      <span>{{ 'COMMON.GOBACK' | translate }}</span>
</button>
```

CONCLUSIONS

La gamificació es un eina que es pot utilitzar en molts entorns, en aquest projecte l'enfocament ha estat l'entorn educatiu, on ha demostrat ser una forma d'incentivar els alumnes per a poder millorar el seu rendiment acadèmic proposant realitzar els estudis utilitzant mecàniques dels jocs. Gràcies a la tecnologia es molt fàcil d'introduir la gamificació a les classes, i n'hi existeix tot un ventall d'aplicacions per fer-ho. Tot i això, encara no hi ha cap aplicació que sigui tan ambiciosa com Classpip i pretengui integrar tants tipus de jocs alhora en una mateixa aplicació.

Conclusió tècnica

L'entorn de Classpip ha crescut molt des dels seus inicis i ha incorporat moltes funcionalitats noves com per exemple les col·leccions, les competicions, etc. Fins ara no hi havia un mètode de treball comú, cada adició al projecte es feia en un repositori nou que després no s'unia a l'original. El resultat va estar que per tal d'unir aquests treballs calia que un estudiant realitzes un projecte d'unió d'aquests altres projectes, i la documentació i manuals acabaven estant molt dispersos. Per aquesta raó, es va desenvolupar un mètode de treball amb git, afegit als annexos, que s'ha utilitzat durant el desenvolupament d'aquest projecte juntament amb altre grup per testejar el seu correcte funcionament, i que un cop acabats els dos projectes el resultat quedi funcional i en un mateix repositori amb èxit.

Degut a que el projecte ha crescut tant, també ha causat que el seu desenvolupament s'hagi prolongat molt en el temps, raó per la que les versions d'Angular, Material i la resta de mòduls utilitzats han quedat molt desactualitzades. Això ha causat que apareguin múltiples avisos de bugs de seguretat cada cop que es fa la instal·lació dels mòduls de node, i una de les pitjors coses es el desfasament entre la versió utilitzada d'Angular i Material i la documentació online disponible. El problema es que a la documentació actual, basada en les ultimes versions d'Angular i Material n'existeixen mètodes i components nous que no existien en les versions antigues utilitzades. Aquesta diferencia entre les versions utilitzades i la documentació pot provocar que s'intentin realitzar sense èxit les implementacions d'elements i/o funcionalitats llistades a la documentació abans d'adonar-se de que no estan disponibles a la versió utilitzada, i fent que finalment s'hagin de desenvolupar amb codi aquestes funcionalitats que només s'haurien d'implementar si s'utilitzés les darreres versions. El cas es que per actualitzar aquestes versions s'han de canviar fins a tres versions majors d'Angular i de Material, cosa que no es recomana per les incompatibilitats entre versions que poden fer que algunes parts del codi deixin de funcionar o funcionin incorrectament.

Per aquesta raó es va crear una branca al repositori anomenada "angular-update", on es va realitzar l'actualització de tots els mòduls que integren el projecte de Classpip a la última versió que hi havia en data 18/09/2018, les principals actualitzacions eren Angular 6.1.7 i Angular Material 6.4.7. Per a fer l'actualització es van haver de fer diversos canvis, ja que al canviar versions majors moltes coses deixaven de funcionar. El codi a aquesta branca funciona

correctament, ja que es van solucionar un per un tots els problemes detectats de compatibilitat i de codi. Tot i això, en realitzar treball en paral·lel amb altres grups d'estudiants, i amb l'objectiu final d'unir tot el codi i mantenir la seva compatibilitat, no es va publicar el desenvolupament realitzat en aquesta branca, ja que podia ser que en unir aquest projecte amb els altres deixessin de funcionar moltes funcionalitats dels altres projectes.

En la mateixa línia, el framework d'Strongloop va aturar el seu desenvolupament ja fa dos anys, el que significa que deixa de tenir actualitzacions, millores i solucions de bugs. Per aquesta raó, com s'ha explicat durant el projecte, hi ha coses com integrar objectes de segon nivell a les consultes de l'API utilitzant filtres que no es poden realitzar, i que no s'afegiran. El recomanable seria canviar Strongloop per un altre framework similar que encara continuï el seu desenvolupament. El problema es que fer aquest canvi pot canviar tots els endpoints de l'API i fer que sigui necessari tornar a reescriure els serveis del dashboard i l'aplicació mòbil.

Conclusió personal

Personalment, portava molt de temps volent aprendre Angular i fer algun desenvolupament amb aquest framework, aquest projecte m'ha donat l'oportunitat de per fi poder aprendre i utilitzar Angular, que a més he ampliat amb coneixements d'Angular Material i fins i tot, encara que finalment no he fet cap desenvolupament utilitzant-lo, he après Ionic. També he tingut l'oportunitat de donar els primers passos en el Sass, que gràcies a que es semblant al Less no m'ha sigut gaire complicat d'aprendre els conceptes bàsics.

He d'admetre que els primers passos de la posada en marxa del projecte van anar molt bé i molt ràpid, encara que donada l'envergadura actual del projecte i la dispersió de la documentació, si em va costar una mica més adaptar-me i comprendre com s'havien fet algunes funcionalitats.

Tot i que vaig aprendre Angular amb tutorials abans de començar a desenvolupar per al projecte, vaig trobar que fins que no estava realitzant tasques per a aquest projecte no havia après ben be com funcionava Angular. I ha estat durant el desenvolupament d'aquestes tasques del projecte on he acabat d'entendre molts conceptes d'Angular que abans no havia acabat comprenent del tot. He après molt durant el desenvolupament de Classpip, de fet, quan gairebé al final del desenvolupament, vaig revisar alguns dels primers codis que vaig realitzar per al projecte, vaig trobar que eren millorables amb els nous coneixements que havia anat adquirint mentre feia el projecte, i no em vaig poder estar de tornar a modificar-los per millorar-los.

Com a desenvolupador d'aplicacions web, estic segur de que aprendre aquestes tecnologies, i guanyar experiència fent-les servir m'obrirà noves possibilitats laborals. Fet pel que estic molt content d'haver iniciat aquest projecte.

Línies obertes

Aquest projecte s'ha centrat en la creació de la nova web de projecció externa i recopilació de material d'onboarding, afegir algunes funcionalitats al projecte, millorar algunes parts del codi i refer completament la home del dashboard per a que sigui més útil i utilitzi les noves funcions desenvolupades.

Però encara queden moltes coses per millorar en aquest projecte, una de les possibles millores es implementar els canvis realitzats al dashboard en l'aplicació mòbil.

També s'hauria d'aplicar l'estructura d'edició i esborrat dels llistats que s'han implementat a les funcionalitats de l'administrador a la resta de llistats del projecte, ja que amb aquest canvi es guanyaria en usabilitat.

El menú de navegació del dashboard no es responsiu, s'hauria de revisar com esta implementat i proporcionar una solució que s'adapti be a les pantalles mòbils.

La diferenciació entre punt i insígnia tal i com s'estaven utilitzant no era gaire clara, raó per la qual es van obviar les insígnies i deixar només els punts en aquest projecte. S'ha de definir bé la diferencia entre punt i insígnia. I restaurar les opcions d'insígnies amb la nova definició.

S'hauria d'implementar un filtre en el llistat d'alumnes que veu el professor a la home en la que el professor pugui triar els alumnes que es mostren al ranking d'alumnes. Si tots els alumnes, o només els alumnes d'algun dels cursos que imparteix aquest professor.

BIBLIOGRAFIA

[1] Universitat Politècnica de Catalunya [online] <https://www.upc.edu/ca> [Consulta: 1 Sep. 2018]

[2] Git Hub [online] <https://github.com/> [Consulta: 1 Sep. 2018]

[3] angular.io [online] <https://angular.io/tutorial> [Consulta: 2 Sep. 2018]

[4] Angular Material [online] <https://material.angular.io/> [Consulta: 4 Sep. 2018]

[5] ionicframework [online] <https://ionicframework.com/docs/v3/intro/tutorial/> [Consulta: 10 Sep. 2018]

[6] StrongLoop [online] <https://strongloop.com/> [Consulta: 16 Sep. 2018]

[7] Node.js [online] <https://nodejs.org/es/> [Consulta: 17 Sep. 2018]

[8] Gamificació [online] <https://www.educaciontrespuntocero.com/noticias/gamificacion-que-es-objetivos/70991.html> [Consulta: 20Sep. 2018 Ma. 2018]

[8] Gamificació [online] <https://www.educaciontrespuntocero.com/recursos/herramientas-gamificacion-educacion/33094.html> [consulta: 20 Sep. 2018]

[9] Gamificació [online] <http://blog.eixestels.com/blog-educativo> [consulta: 20 Sep. 2018]

[10] Kahoot [online] <https://kahoot.com/> [Consulta: 21 Sep. 2018]

[11] Knowre [online] <http://knowre.com/> [Consulta: 21 Sep. 2018]

[12] Classdojo [online] <https://www.classdojo.com/es-es/> [Consulta: 21 Sep. 2018]

[13] Brainscape [online] <https://www.brainscape.com/> [Consulta: 21 Sep. 2018]

[14] Classbadges [online] <http://classbadges.com> [Consulta: 21 Sep. 2018]

[15] CodeCombat [online] <https://codecombat.com> [Consulta 21 Sep. 2018]

[16] Prismjs [online] <https://prismjs.com/> [Consulta 8 Nov. 2018]

CAPÍTOL 6. ANNEXOS

6.1 Tutorial Git amb mètode de treball en grup

6.1.1 Introducció a Git

Git és un sistema de control de versions distribuït. Git és una multiplataforma, per el que es pot utilitzar sense restriccions amb els sistemes operatius més comuns; com són Windows, Linux o MacOs.

Gràcies a Git es van guardant els canvis que es realitzen en els diversos arxius del projecte, podent anar enrere o cap endavant amb unes simples ordres. Algunes de les funcions més útils de Git són:

- Poder comparar el codi d'un arxiu amb les seves versions anteriors, per veure els canvis realitzats en aquest.
- Restaurar versions anteriors d'un arxiu.
- Unir els canvis en un arxiu amb els canvis realitzats per una altra persona.
- Dividir el projecte per branques, per a que els canvis realitzats en una de les seves branques no afecti als altres.
- Pujar els canvis realitzats a un repositori remot per a que estiguin disponibles per a la resta de l'equip.

Per al projecte de Classpip s'ha utilitzat Github, que és un servei d'allotjament de repositoris gestionats amb Git.

Si no s'està familiaritzat amb Git, hi han varis recursos amb els que aprendre Git de forma fàcil i pràctica.

https://www.codecademy.com/learn/learn-git
https://www.katacoda.com/courses/git
https://learngitbranching.js.org

Hi ha força coses que aprendre amb Git, però per fer-ho servir correctament amb el projecte, és important familiaritzar-se amb els conceptes de:

- Afegir i comitejar canvis (git add i git commit).
- Moure entre versions (git checkout).
- El concepte de branca, poder veure-les, crear-les i canviar entre elles (git branch, git checkout).
- Fusionar els canvis realitzats en una branca amb altra (git merge).

6.1.2 Els repositoris de Classpip

Els repositoris de Classpip es troben en Github.com, els seus enllaços són els següents:

https://github.com/alejandromartincruz/classpip-services
https://github.com/alejandromartincruz/classpip-dashboard
https://github.com/alejandromartincruz/classpip-mobile
https://github.com/alejandromartincruz/classpip-onboarding

6.1.3 Primeres passes amb repositori Classpip

Per començar a treballar amb un dels projectes, és suficient amb clonar el projecte amb el nostre equip. Per això, n'hi haurà prou en obrir el terminal, anar a la carpeta on es vulguin tenir els projectes i utilitzar la següent ordre:

```
git clone <url-del-repositorio>
ejemplo: git clone https://github.com/alejandromartincruz/classpip-mobile.git
```

Una vegada clonat el projecte amb l'equip, es procedirà a entrar a la seva carpeta corresponent, i des d'allà es pot verure l'estat (git status), les branques del projecte (git branch) i les seves versions (git tag). En estar acabat de baixar l'estat mostrarà que no hi ha canvis, i només mostrarà la branca master.

Els repositoris tenen la branca master, la dev, la test i la tutorial, Per obtenir una branca del repositori és tan senzill com utilitzar les següents ordres:

```
git fetch origin <nombre-rama>
git checkout <nombre-rama>
ejemplo: git fetch origin tutorial
         git checkout tutorial
```

Després d'això, tot el codi del projecte es canviarà per mostrar el codi que hi ha a la branca tutorial. On es veuran els arxius afegits amb el tutorial de desenvolupament per classpip.

Quan es canviï d'una versió a altre dins del repositori, és bastant probable que els moduls de node instal·lats hagin canviat de versió, no s'hagin instal·lat en aquesta versió, o s'hagin insta·lat moduls nous que no estaven a la versió prèvia. Per això, es recomanable esborrar la carpeta node_modules i tornar a realitzar la instal·lació de mòduls.

```
rm –rf node_modules
npm install
```

6.1.4 Protocol de treball en grup a Git

La utilització d'un programa de control de versions, com git en aquest cas, suposa un gran avanç a l'hora de realitzar treball en equip en qualsevol software.

És convenient marcar unes pautes de treball en equip per facilitar la coordinació de grups de treball en el mateix projecte, agilitzant d'aquesta manera el treball de diverses característiques o mòduls, i la seva posterior integració en el projecte.

Es parteix de la premissa de que actualment hi ha dues branques, la **master** i la **dev**. Tenint això en compte, cada projecte nou començarà amb una nova branca que sorgirà de la versió actual de master. L'objectiu és tenir les branques master i dev sense diferències entre elles, per poder començar a treballar òptimament, i una branca on es realitzarà el projecte que serà idèntica a master en el moment de començar-lo. Cada projecte tindrà un nom únic e identificatiu per branca. A partir d'ara ja es pot considerar:

- La branca master amb la branca de producció, és a dir, la que es veurà públicament i utilitzaran els usuaris.
- La branca dev com la branca de preproducció, una branca serà idèntica a la de producció per provar els desenvolupament abans de que siguin publicats en producció.
- La branca pròpia de cada projecte amb la branca on cada estudiant o grup d'estudiants aniran integrant els diferents desenvolupament del seu projecte.

A més, es tindran en compte dues situacions diferents, el treball de l'equip envoltat en el projecte directament, i el treball per part de persones o equips aliens al projecte però que volen participar.

6.1.5 Cas 1, treball en l'equip directament relacionat amb el projecte:

Primer de tot cal fixar un temps de desenvolupament, unes funcionalitats a desenvolupar o ambdues coses simultàniament seguint una metodologia scrum. Una vegada passat el temps fixat, es publicaran els desenvolupament que s'hagin realitzat correctament durant aquest període de temps. O en el cas d'haver fixat funcionalitats en comptes de temps, es publicaran les funcionalitats una vegada acabades. L'elecció d'un mètode o un altre serà elegit per l'equip de treball.

El següent pas és clonar el projecte, i preparar-lo amb branques **master** i **test**. Quan ja estiguin les dues branques preparades, només falta crear una nova branca per al projecte. És important estar situat en la branca master quan es generi una branca nova per a que aquesta sorgeixi de master. La creació d'aquesta branca es fa mitjançant l'ordre:

```
git checkout master
git checkout –b <nombre-nueva-rama>
```

Una vegada es tingui preparat l'entorn de treball amb les tres branques en la mateixa situació, es podrà començar una metodologia de treball en equip que faciliti la coordinació entre els membres de l'equip, que asseguri el correcte testeig dels treballs individuals de cada un.

Per començar un nou desenvolupament es deurà crear una nova branca a partir de la branca master, que anomenarem a mode d'exemple desarrollo_1. Amb l'ordre:

```
git checkout master
git checkout –b desarrollo_1
```

En aquest moment la branca desarrollo_1 és exactament igual que la branca master, dev i la del projecte. I aquí és on es comença a desenvolupar la nova funcionalitat que es vulgui afegir al projecte.

No s'hauria de fer cap commit fins que aquest nou desenvolupament no estigui finalitzat i funcionant. Si es realitzen molts commits, en el cas de tenir que realitzar una revisió del treball per trobar un bug o per millorar alguna cosa, s'hauran de revisar els canvis realitzats en cada un dels commits individualment, el que dificulta bastant la revisió. Tenir tot el treball realitzat en un sol commit facilitarà la posterior revisió dels canvis realitzats, al tenir-los tots junts.

Una vegada finalitzat el nou desenvolupament, es realitzarà un merge contra la branca del projecte. I es realitzaran les proves per comprovar el correcte funcionament del projecte.

Per realitzar el merge sobre la branca del projecte (anomenada a partir d'ara **proyecto** a mode d'exemple), es farà de la següent manera:

1. `git checkout proyecto`
2. `git pull origin proyecto`
3. `merge --no-ff --no-commit desarrollo_1`
4. `git commit -m "[proyecto] merge with desarrollo_1"`

S'ha de destacar, que abans de realitzar el merge, és important fer un pull com s'indica en el pas 2, en el que es descarregarien tots els possibles canvis realitzats a la branca per un altre membre del grup, i amb el que es realitzaria el merge sobre una branca actualitzada.

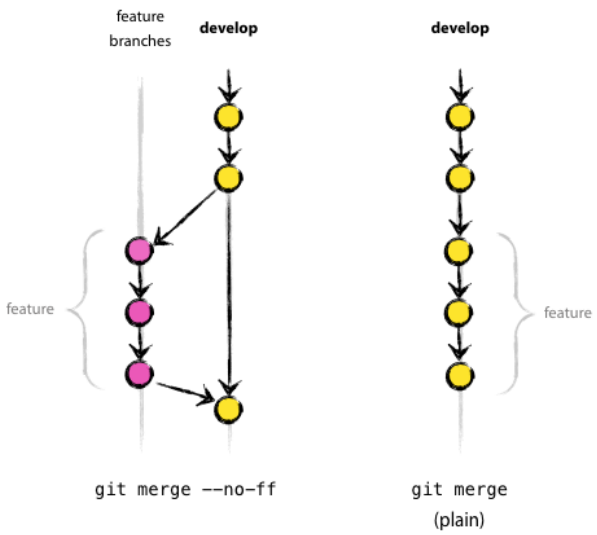
Quan el desenvolupament està correctament finalitzat, i funciona bé el projecte, es realitza un merge sobre dev. El procés de merge en dev és el mateix que en proyecto:

1. `git checkout dev`
2. `git pull origin dev`
3. `merge --no-ff --no-commit desarrollo_1`
4. `git commit -m "[dev] merge with desarrollo_1"`

En cas d'haver un conflicte entre els passos 3 i 4, aquest no quedarà commitejat gràcies al flag "—no—commit", llavors n'hi haurà prou en solucionar el conflicte, afegir els canvis amb un "git add" i commitejar segons el pas 4:

```
1. Solucionar conflicte
2. git add –A
3. git commit –m “[dev] merge with desarrollo_1”
```

El flag “--no-ff” s’utilitza per evitar que és realitzin un fast-forward que ens farà perdre informació de l’històric de branques. En la següent imatge és pot apreciar la diferencia entre realitzar un commit amb el flag i sense el flag.



Ilustració 19: font https://nvie.com/posts/a-successful-git-branching-model/

Si tots els desenvolupaments mergejats a la branca dev funcionen correctament una vegada finalitzat el temps de desenvolupament, o tots els desenvolupaments que és vulguin publicar, es realitzen un merge contra la branca master, arrossegant tots els canvis mergejats contra la branca dev a la branca master, una vegada realitzat aquest merge s’obté una nova versió funcional del projecte, per al que es procedirà a etiquetar dita versió segons el procediment habitual.

El procediment habitual d’etiquetatje és: x.y.z, on:

- X és la versió major, s’utilitza quan hi ha un canvi important funcionant.
- Y es la versió menor, s’utilitza per afegits de funcionalitat que suposen canvis molt grans.
- Z s’utilitza per petites correccions d’errors.

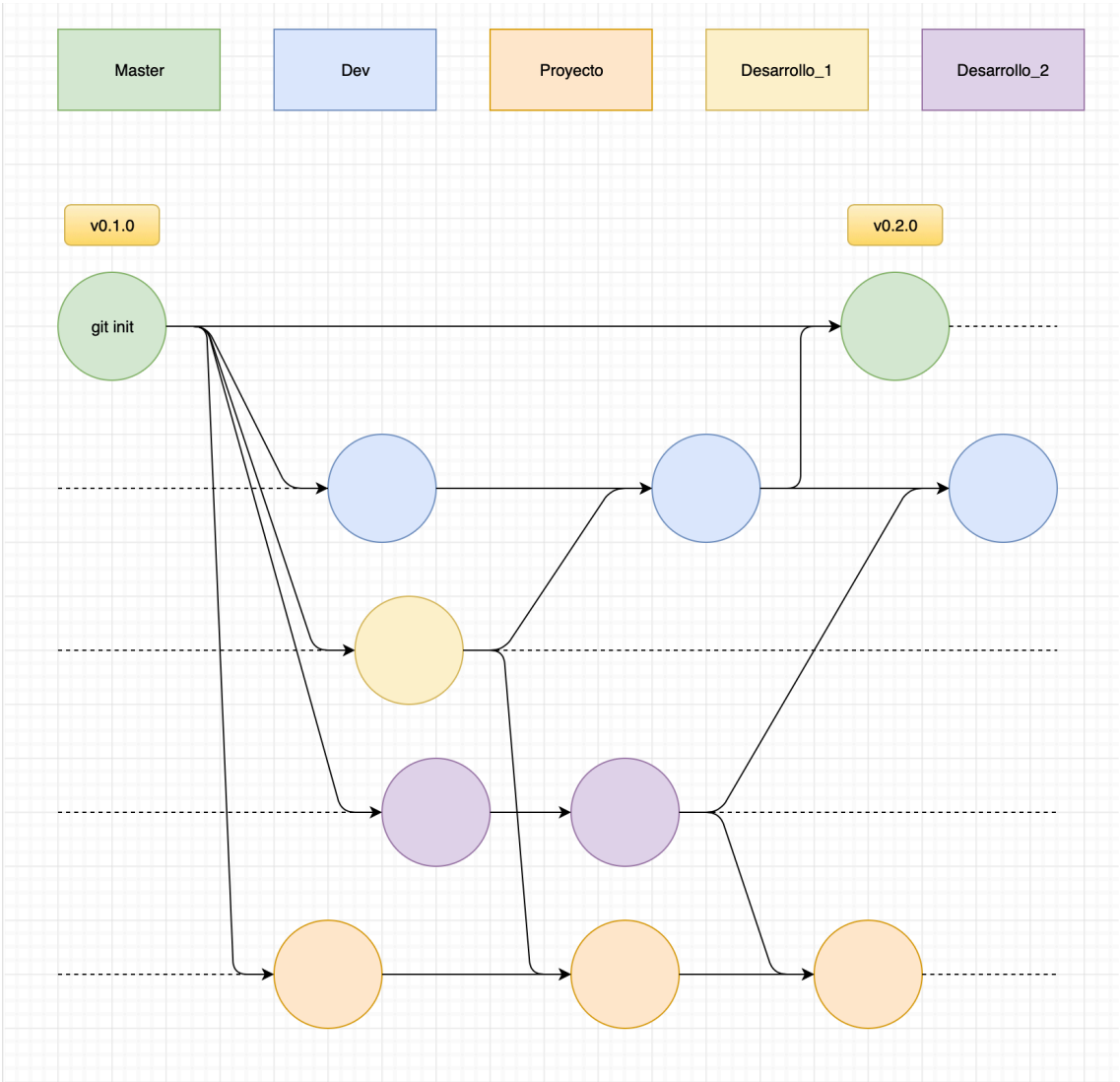
De manera que, una primera versió podria considerar-se com v0.1.0, i després d’afegir una funcionalitat passaria a ser la v0.2.0 després fer canvis importants en l’aplicació com per considerar que ja està llesta, passaríem a denominar-la amb l’etiqueta v1.0.0, i en el cas de que aquesta versió tingués una bug que requereixi d’un fix, la nova versió seria la v1.0.1. Posar un tag a la branca activa és tan fàcil com utilitzar l’ordre:

```
git tag v0.1.0
```

Si alguna cosa funcionés malament a la nova versió, seria molt fàcil revertir els canvis a la versió anterior que va ser etiquetada i que sabem que funcionava bé. Es pot revertir a un estat anterior utilitzant els tags en lloc dels commits, l'ordre per realitzar aquest canvi:

```
git checkout v0.1.0
```

Utilitzant l'ordre *"git tag"* es mostra un llistat de versions per etiqueta, que si s'ha seguit el protocol de numeració correctament serà fàcil determinar quina és la versió prèvia a l'actual.



Ilustració 20: exemple de línia de treball

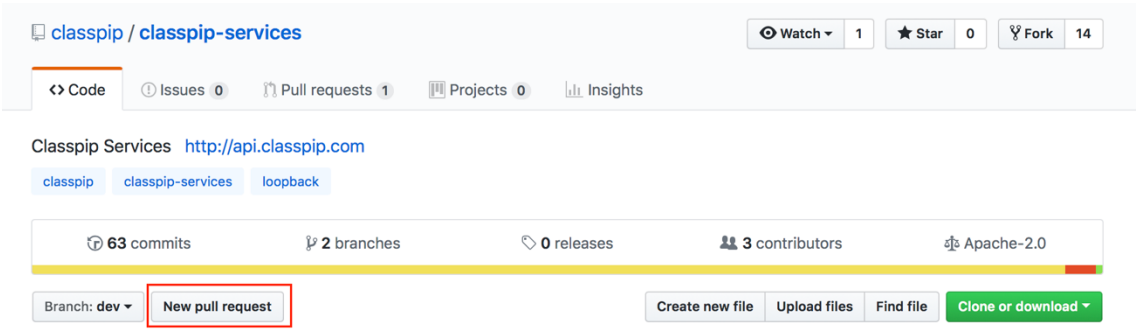
6.1.6 Cas 2, aportacions de persones o equips aliens al projecte

És de sobres conegut que un projecte open source acaba sent tan potent com la comunitat de desenvolupadors que hi ha darrere. Sent les col·laboracions

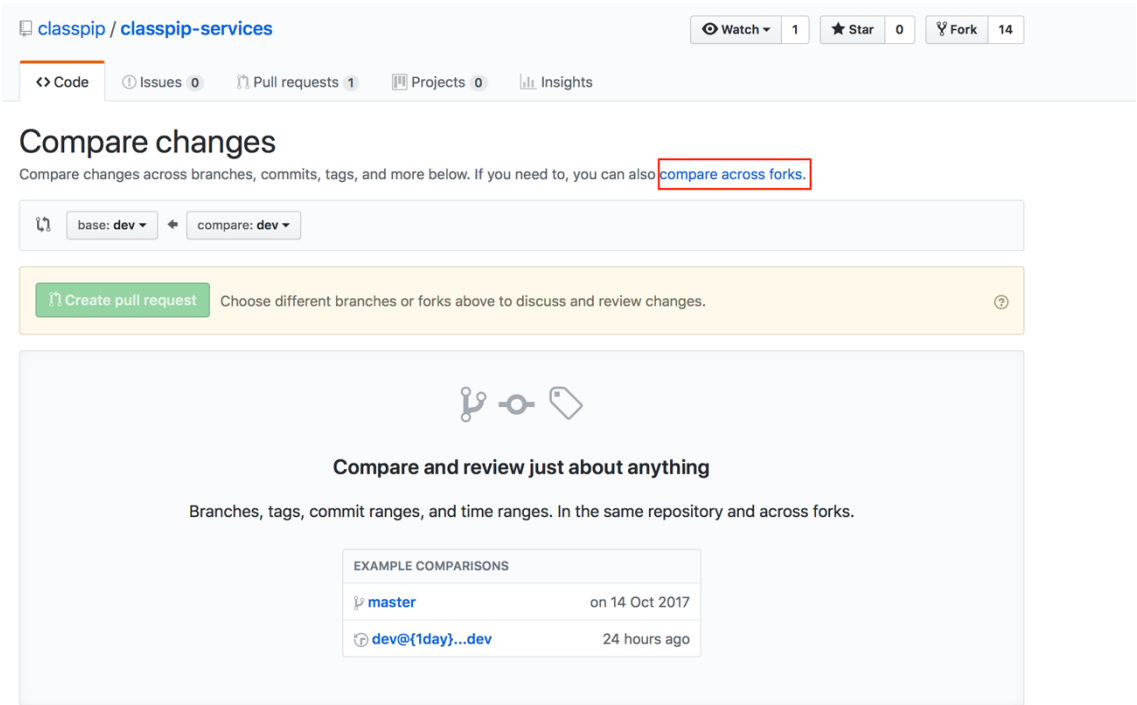
d'aquets tan o més importants com les de l'equip de treball directament implicat. És per això que cal plantejar un mètode de treball per publicar en el projecte les aportacions de tercers.

En aquest cas els desenvolupadors necessitaran crear un fork del repositori original, passant a tenir el seu propi repositori i el seu propi remot. La forma de treballar deu ser semblant a la del cas 1, amb l'excepció de que per publicar algo deuran fer un pull request.

Per fer el pull request s'ha d'anar al repositori original des del que es va realitzar el fork, i prémer el botó "New pull request".

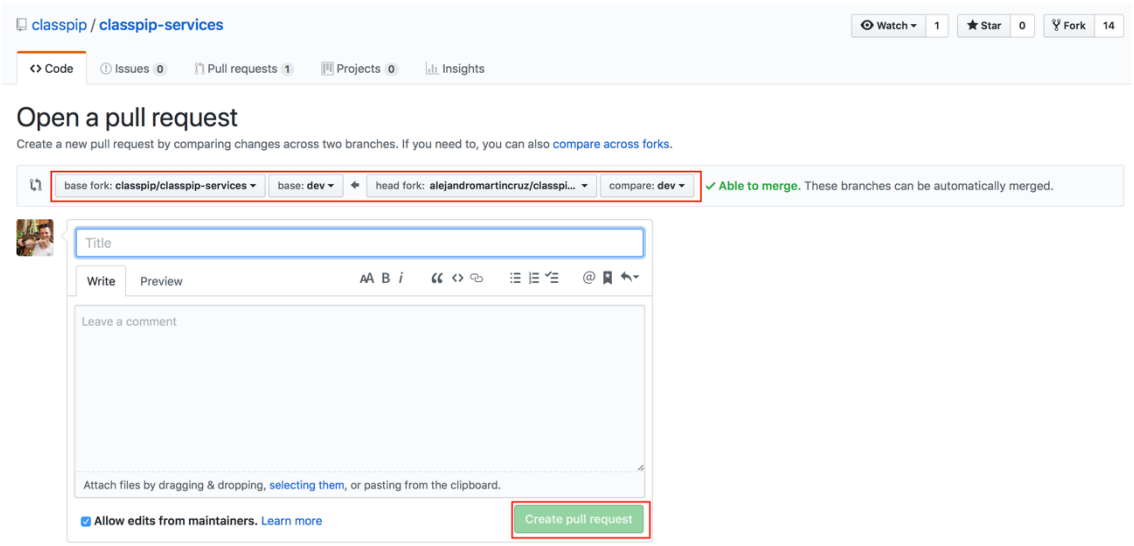


En aquest punt, es mostra una pàgina amb el títol "Compare changes", en aquesta pàgina s'ha de prémer sobre l'enllaç *"compare across forks"*.



A continuació, s'ha de seleccionar com "base fork" la branca dev del repositori original, i com "head fork" la branca dev del repositori del col·laborador. A més,

s'haurà d'escriure un títol i una descripció per facilitar la comprensió dels canvis en el codi a qui deu aprovar el pull request. I ja es podrà prémer la sol·licitud de pull request.



Ara un membre de l'equip amb accés al repositori original podrà aprovar, rebutjar o sol·licitar una revisió en el pull request.

El membre de l'equip, amb permisos d'escriptura al repositori, deurà anar a la pestanya "pull requests" que apareixerà ara al projecte, seleccionar el pull request que es vulgui integrar en el projecte, i una vegada revisat el codi, per acceptar-lo només tindrà que seleccionar *"aprove"* i després prémer el botó *"Submit review"*.

Les altres dues opcions que hi ha per seleccionar en un pull request són *"Comment"*, on el revisor pot posar un comentari sobre el pull request sense haver d'aprovar-lo.

I *"Request changes"*, on és sol·licita que es realitzi canvis en el codi que es vulgui mergejar al repositori.

6.1.7 Consell tutorial de desenvolupament mòdul de Mesa

Per seguir el tutorial de desenvolupament en el que es realitza un mòdul de mesa d'estudiant, és important tenir en compte que s'ha de situar en el tag v0.1.0 abans de començar a intentar desenvolupar el mòdul del tutorial. Doncs el codi del tutorial parteix d'aquest punt, i s'iniciarà el desenvolupament en un altre punt, el codi seria diferent i podria portar a errors o confusions.

Si en algun moment es vol comparar el codi que s'està realitzant amb el codi del tutorial, existeixen dues opcions. La primera es anar-hi al repositori de Github del

projecte, un cop allà situar-se a la branca tutorial i cercar l'arxiu que es vol comparar.

El segon mètode es pot executar directament des de la línia de comandes, i com a resultat mostrarà les diferencies entre l'arxiu que s'està editant i l'arxiu del tutorial. D'aquesta manera es pot veure clarament que es exactament el que resta per a realitzar les passes indicades al tutorial. Per a veure aquesta diferencia, s'ha d'utilitzar la comanda *"git diff"* de la següent manera:

```
git diff master..tutorial -- myfile.ts
```

Aquesta comanda suposa que s'està realitzant el tutorial des de la branca master, en cas d'estar realitzant-se des de una branca personalitzada iniciada en master al punt v0.1.0, s'haurà d'utilitzar el nom d'aquesta branca en comptes de master. En lloc de myfile.ts s'ha de posar el nom de l'arxiu que es vol comparar.

6.2 Manual de desenvolupament de la web d'onboarding

6.2.1 Afegir pàgina a la web d'onboarding

Per afegir una pagina a la web d'onboarding, primer s'ha de crear el nou component que allotjarà el text, les imatges i la resta de components d'aquesta pàgina. En aquest exemple es crearà una nova pàgina de privacitat. Per a crear un nou component s'ha d'utilitzar el terminal. Un cop obert el terminal, primer s'ha d'anar a la carpeta del projecte, i després s'ha d'utilitzar la comanda:

```
ng generate component shared/privacy
```

Aquesta comanda, ha generat la carpeta privacy dins de app/shared, i al mateix temps dins de la carpeta privacy es generen quatre arxius:

- **privacy.component.html:** aquí es on s'escriu el codi html amb el contingut del component (en aquest cas pagina).
- **privacy.component.scss:** aquí es on s'escriu el codi css en formato sass específic per aquest component.
- **privacy.component.spec.ts:** en aquest arxiu s'escriuen els test del codi, si es que s'utilitzen.
- **privacy.component.ts:** en aquest arxiu s'escriuen les importacions necessàries per al component, el codi que s'iniciarà al carregar la pagina, les variables que estaran disponibles a la pagina, etc.

Gracies a aquesta comanda, també s'ha afegit el component en app.module.ts, tant la importació com la declaració.

El següent que es necessita es generar la ruta per a aquesta nova pagina, per fer això s'ha d'anar a l'arxiu "app.routing.ts", importar el component que s'ha generat i afegir-li la ruta.

```
import { PrivacyComponent } from
'./shared/privacy/privacy.component';

import { NotfoundComponent } from
'./pages/notfound/notfound.component';

const routes: Routes = [

  { path: 'terms/privacy',    component: PrivacyComponent },

  { path: '404', component: NotfoundComponent },

  { path: '**', redirectTo: '/404' }

];
```

En aquest exemple s'aprecia que s'afegeix PrivacyComponent al costat de la ruta NotfoundComponent i una ruta per defecte que porta qualsevol ruta no declarada al component de notfound. D'aquesta manera queda clar com afegir noves rutes a les ja existents.

Un cop ja es tenen el component i la ruta, ja es poden crear els enllaços que porten cap a aquesta ruta, o fins i tot escriure-la directament al navegador. Ara ja només resta escriure el contingut a l'arxiu privacy.component.html.

6.2.2 Afegir enllaços al menú per a noves pàgines

Per a modificar els enllaços del menú sidenav s'ha d'anar a l'arxiu src/app/shared/navigation/links/links.component.html, en aquest arxiu es troba un mat-nav-list que conté els enllaços del menú.

Existeixen dos nivells d'enllaços, el que va directament en el menú sidenav (primer nivell), i els que es veuen quan aquests s'obren (segon nivell). Tots els enllaços han de portar mat-list-item dins de l'html tag a (anchor).

A tots els enllaços que porten a una pagina nova se'ls ha de posar la funció settitle('títol de la nova pàgina') per a que quan es dispari l'event click realitzi aquesta funció. Això es fa per a que el títol de la pestanya canviï i cada pàgina tingui el seu propi títol. setTitle es una funció d'un servei d'Angular creada per a canviar el títol de les pàgines. Que cada pàgina tingui el seu propi títol millora el SEO (Search Engine Optimization) de la web, i per tant, també la seva posició a l'hora d'aparèixer en els resultats de cercadors com Google.

Als enllaços de primer nivell, si contenen enllaços de segon nivell, se'ls ha de posar la funció openClose('idGrupo', 'idGrupoCaret'), i dins del contingut del tag anchor s'ha de posar una icona caret de Material amb un id únic que s'utilitzarà

per assignar a cada grup d'enllaços i que es el paràmetre 'idGrupoCaret' de la funció. L'altre paràmetre de la funció es el nom d'una variable booleana que s'utilitzarà per a mostrar o amagar els enllaços de segon nivell d'aquest grup. La funció retorna el valor true o false que s'assigna a aquesta mateixa variable i gira la icona amb el id proporcionat en el segon paràmetre.

```
<a mat-list-item (click)="usersShow = openClose(usersShow,
'userCaret')">

    <mat-icon mat-list-icon>people</mat-icon>

    <span>Usuarios</span>  // titulo enlace

    <span class="fill-space"></span>  // span vacio para alinear el
caret a la derecha

    <mat-icon id="userCaret" class="material-
icons">expand_more</mat-icon>

</a>
```

Els enllaços de segon nivell han de tenir un *ngIf="idGrupo", que al ser idGrupo una variable booleana els enllaços es mostraran o amagaran segons el seu valor. A aquests enllaços de segon nivell també se'ls ha d'afegir la funció openClose() en l'event click, per a que es tanqui el grup un cop seleccionat un dels enllaços.

```
<a *ngIf="usersShow" class="second-level" mat-list-item
routerLink="/user/students" (click)="usersShow =
openClose(usersShow, 'userCaret'); setTitle( 'Classpip onboarding
aprende a usar la aplicacion como estudiante' );">

    <mat-icon mat-list-icon fontSet="fa" fontIcon="fa-user-
graduate"></mat-icon>  // ver sección iconos

    <span>Estudiantes</span>  // titulo enlace

</a>
```

En l'exemple previ, s'observa con esta realitzat l'enllaç de primer nivell "Usuarios", i un enllaç de segon nivell que pertany a aquest i que s'anomena "Estudiantes".

6.2.3 Icones

En aquest projecte es poden utilitzar els material icons de la forma usual, que es <mat-icon>nomIcona</mat-icon>, utilitzant com a "nomIcona" qualsevol dels de la llista de icones de material disponibles.

Però també s'han afegit les icones gratuïtes de font Awesome, que ofereixen una major varietat de icones que no existeixen en les icones de material, com per exemple la icona de Github. Per a poder utilitzar les icones de Font Awesome dins d'un tag mat-icon, com els de material, s'ha instal·lat un mòdul de node i s'ha creat un servei.

Ara, per a afegir una icona de Font Awesome només cal afegir al tag de mat-icon un fontSet i un fontIcon. El fontSet depèn del tipus d'icona que es vulgui utilitzar, les normals utilitzen el fontSet fa, les de marques utilitzen el fab, les del grup sòlid utilitzant el fas, etc. En el fontIcon s'ha de posar el nom de la icona precedit de fa-. I finalment, al contrari que amb les icones de material, no s'ha de posar cap text dins del tag mat-icon.

Si per exemple es vol utilitzar la icona address-card del grup sòlid, s'ha de fer de la següent manera:

```
<mat-icon mat-list-icon fontSet="fas" fontIcon="fa-address-
card"></mat-icon>
```

6.2.4 Crear un nou servei

Com exemple de creació de servei, es crearà el servei que dona estil al format de codi que hi ha a les pagines de desenvolupament del material d'onboarding de la web. Per a això s'utilitzarà el mòdul anomenat prismjs. Per tant, el primer pas es instal·lar aquest mòdul introduint la següent comanda en el terminal situant-se a la mateixa carpeta del projecte.

```
$ npm install prismjs --save
```

Ara s'ha de crear el nou servei. Per a generar-lo s'ha d'introduir la següent comanda del cli d'Angular:

```
$ ng generate service shared/services/highlight
```

El servei que s'ha generat, podem trobar-ho en la ruta app/shared/services/highlight.service.ts, i de moment només te codi genèric. L'arxiu ha de contenir el següent:

```
import { Injectable } from '@angular/core';

@Injectable({
  providedIn: 'root'
})
export class HighlightService {

  constructor() { }

}
```

Es procedeix a escriure el codi del servei, En aquest cas, s'escriu codi per a poder utilitzar prismjs i que aquest doni format de codi, amb colors per a facilitar la lectura dels fragments de codi que hi ha inserits als manuals del material d'onboarding de la web. Un cop escrit el codi, l'arxiu highlight.service.ts quedarà de la següent manera:

```
import { , PLATFORM_ID, Inject } from '@angular/core';

import 'clipboard';
import 'prismjs';
import 'prismjs/plugins/toolbar/prism-toolbar';
import 'prismjs/plugins/copy-to-clipboard/prism-copy-to-clipboard';
import 'prismjs/components/prism-javascript';
import 'prismjs/components/prism-markup';
import 'prismjs/components/prism-typescript';
import 'prismjs/components/prism-sass';
import 'prismjs/components/prism-scss';

declare var Prism: any;

@Injectable({
  providedIn: 'root'
})
export class HighlightService {
  constructor(@Inject(PLATFORM_ID) private platformId: Object){}
  highlightAll() {
    if (isPlatformBrowser(this.platformId)) {
      Prism.highlightAll();
    }
  }
}
```

Un cop programat el servei que es necessita, s'ha d'utilitzar en el component on es vulgui inserir codi formatat. Com exemple, es va a inserir el codi de formatat de codi en el component anomenat onboarding. En aquest cas concret, a més s'utilitzarà AfterViewChecked, que es un event que es llença quan la vista ja s'ha carregat completament, i es quan s'iniciarà el mètode highlightAll(). També s'ha afegit la variable booleana highlighted que s'utilitza per a evitar que el mètode highlightAll() es cridi més d'un cop. Per això, després d'importar AfterViewChecked de Angular/core i el servei HighlightService, s'ha de afegir private highlightService_ HilghlightService al constructor, i per últim, la funció que es llença en ngAferViewChecked. Una vegada realitzat això el codi ha de ser com aquest:

```
import { Component, OnInit, AfterViewChecked } from
'@angular/core';

import { HighlightService } from
'../../shared/services/highlight.service';

@Component({

  selector: 'app-onboarding',

  templateUrl: './onboarding.component.html',

  styleUrls: ['./onboarding.component.scss']

})

export class OnboardingComponent implements OnInit,
AfterViewChecked {

  onboarding: OnboardingInterface;

  highlighted: boolean = false;

  constructor(private highlightService: HighlightService) { }

  ngAfterViewChecked() {

    if (this.onboarding && !this.highlighted) {

      this.highlightService.highlightAll();

      this.highlighted = true;

    }

  }

}
```

En aquest cas, per acabar encara resta un últim pas. S'han d'importar els fulls d'estils de prismjs a src/sass/styles.scss. Que en el cas d'aquest projecte, son les que pertanyen al tema *coi*. I de ser necessari, després es pot afegir una personalització al codi per a que s'adapti a l'estil de la web. La importació i personalització d'estils es la següent:

```
@import "~prismjs/plugins/toolbar/prism-toolbar.css";

@import "~prismjs/themes/prism-coy";

pre[class*="language-"] {

  background-color: transparent;

}

pre[class*="language-"]:before, pre[class*="language-"]:after {

  display: none;

}
```

El servei ja esta afegit i funcionant. Ja s'ha explicat com crear un nou servei i com utilitzar-lo en els components. Ara cal remarcar per a utilitzar el formatat de text del servei de highlight, només s'ha d'afegir la classe class="language-llenguatge-utilitzat" al tag code, per exemple, per a codi escrit en javascript s'ha d'afegir class="language-typescript" al tag code. Encara que, aquest formatat de codi, esta explicat més àmpliament al punt 6.2.6.

6.2.5 Afegir o modificar dades en pàgina de Professors

Las pàgines del projecte d'onboarding estan fetes amb contingut estàtic, a excepció de la pàgina amb el tutorial d'us per a professors que te un arxiu amb el contingut que s'importa i es llista des de els components corresponents a la pàgina de professors.

Per a aquest contingut, s'ha creat una classe dataCard en app/class/dataCard.ts que s'utilitza en l'arxiu app7data/teacher_videos.ts, on s'ha escrit un array d'objectes dataCard.

L'ultim que resta es importar la classe dataCard i l'array d'objectes dataCard en el component app/pages/teachers/teacher.component.ts i igualar l'array a una variable que el pugui utilitzar a la plantilla i utilitzar per a llistar el seu contingut.

S'ha de mencionar que per a que es puguin llistar enllaçosa webs provinents d'objectes, com ho son els enllaços de Youtube, s'ha de "sanejar" el contingut, ja que si no es fa, aquest queda bloquejat per seguretat. Per a sanejar-lo s'ha d'importar DomSanitizer de angular/platform-browser i introduir-ho en el

constructor. Ara que ja es poden utilitzar els mètodes de sanitizer, allà on es vagi a imprimir el contingut url s'ha de posar el següent: [src]='sanitizer.bypassSecurityTrustResourceUrl(url)'.

El codi del component de teachers.component.ts es el següent:

```
import { Component, OnInit } from '@angular/core';

import { DomSanitizer } from '@angular/platform-browser';

import {CdkDragDrop, moveItemInArray} from '@angular/cdk/drag-
drop';

import { DataCard } from '../../class/dataCard';

import { CARDS } from '../../data/teacher_videos';

@Component({

  selector: 'app-teachers',

  templateUrl: './teachers.component.html',

  styleUrls: ['./teachers.component.scss']

})


export class TeachersComponent implements OnInit {


  cards = CARDS;

  constructor(public sanitizer: DomSanitizer) {}

  ngOnInit() {}


  drop(event: CdkDragDrop<DataCard[]>) {

    moveItemInArray(this.cards, event.previousIndex,
event.currentIndex);

  }

}
```

6.2.6 Inserir fragments de codi

Per a facilitar la lectura de codi en els tutorials de la web d'onboarding s'ha utilitzat el mòdul de Node prismjs, i com s'ha vist en un punt anterior d'aquest tutorial s'ha creat un servei per al seu ús.

Per utilitzar el formatat de codi en un component es fa de la següent manera:

- S'importa el servei en el component "import { HighlightService } from 'ruta-fins/shared/services/highlight.service';"
- S'afegeix "private highlightService: HighlightService" al constructor de la classe.
- S'afegeix la variable "highlighted: boolean = false;"
- S'afegeix "AfterViewChecked" després del "implements OnInit"
- S'afegeix el mètode "ngAfterViewChecked()"

```
ngAfterViewChecked() {
        if (this.onboarding && !this.highlighted) {
          this.highlightService.highlightAll();
          this.highlighted = true;
        }
}
```

En aquest cas, es pot seguir d'exemple el codi que hi ha a l'anterior apartat. On s'implementa el servei de highlightService en el component de teachers.component.ts

En el cas d'utilitzar tabs, aquests utilitzen lazy loading, el que significa que van carregant contingut quan s'obre el corresponent tab, i no abans. El servei de hightlightService esta implementat de forma que nomes carrega un cop quan carrega la pàgina. D'aquesta manera, quan acaba de carregar la pàgina es dona format al codi, però al canviar el tab, carrega el nou contingut y no es llença la funció de formatat, per el que es veu el text sense format. Per solucionar-h, s'ha d'utilitzar el event selectedTabChange per llençar la funció tabSelectionChanged($event) en el tag mat-tab-group.

```
<mat-tab-group (selectedTabChange)="tabSelectionChanged($event)">
```

La funció tabSelectionChanged(event) el qe fa es tornar a llençar el servei de highlightService per a el nou. La funció es la següent:

```
tabSelectionChanged(event) {

    this.highlightService.highlightAll();

}
```

Una vegada que esta tot a punt per formatar el codi, només cal afegir a la plantilla html del component. Per a això s'utilitzen els tags code i pre.

En el tag pre es pot seleccionar si mostrar o no las línies en el codi. Per mostrar les línies s'ha d'afegir la classe line-numbers al tag pre, en el cas de posar les línies del codi es pot triar en que número començar. Si no es posa res, comença per la línia número 1, però si es vol començar per alguna línia en concret s'ha de posar un data-start amb el número de línia per el que es vol començar. Si, per exemple, es volen posar números y començar per la línia 17, el tag pre quedarà de la següent manera <pre class="line-numbers" data-start="17">.

L'únic que queda es indicar que tipus de codi es va a mostrar per a que li faci un formatat correcte. Per això s'ha d'afegir una classe al tag code, la classe es language-tipus-llenguatge, per el que si es va a mostrar llenguatge typescript s'ha d'afegir class="language-typescript".

Ara mateix, els llenguatges que es poden formatar son els importats en l'arxiu highlight.services.ts, y son:

- javascript
- markup (actualment utilitzat per mostrar html)
- typescript
- sass
- scss

Encara que es poden afegir qualsevol dels llenguatges disponibles a la pagina de prismjs. Nomes s'han afegit aquests llenguatges, ja que, de moment, son els únics utilitzats en el projecte.

Important: Angular compila el codi que hi ha dins dels pre-code, y normalment dona error al introduir codi y compilar. La forma de solucionar-ho es la següent:

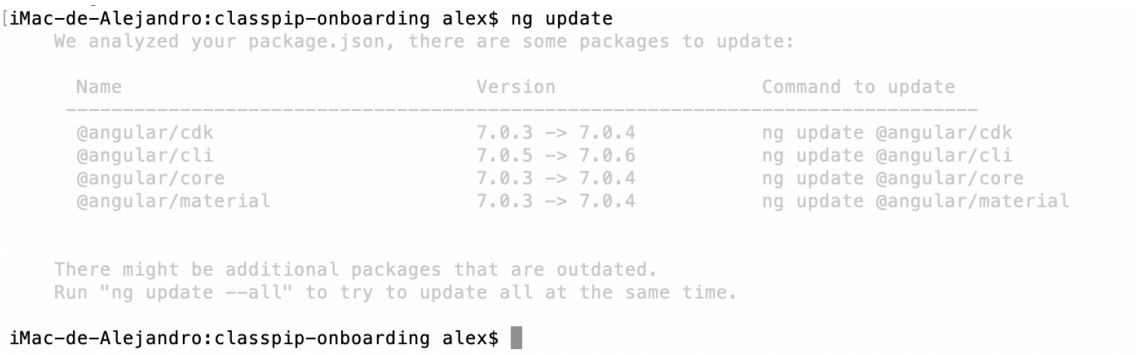
- Si apareix algun caràcter *{* y *}* en el codi s'ha de posar *{{'* al principi del codi, i *'}}* al final.
- En ocasions el codi te cometes en diversos llocs y la forma anterior no funciona. En aquests casos s'ha de substituir cada *}* per *{{'}'}}* y també s'ha de substituir cada *}* per *{{'}'}}*.
- Si es vol mostrar codi html s'ha de substituir els < i els > dels tags per *<* i per *>* de no fer-ho angular dona error i no carrega la pàgina.

6.2.7 Actualitzar projecte d'onboarding

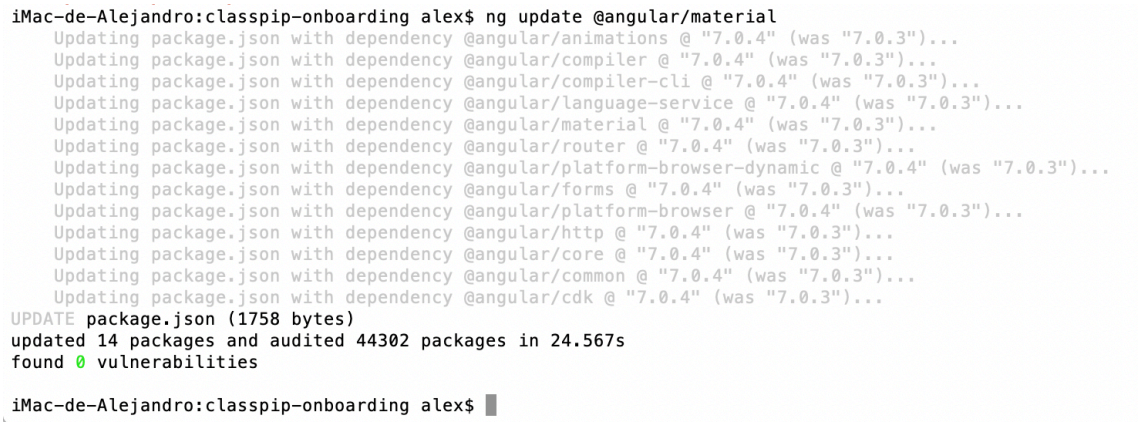
Des de Angular 6 s'ha introduït una forma d'actualitzar la aplicació i les seves dependències molt fàcil y senzilla. Aquest projecte es va iniciar amb Angular 6, i posteriorment s'ha actualitzat en diverses ocasions fins a la versió 7 de Angular. Per actualitzar basta amb introduir la següent comanda en el terminal:

```
ng update
```

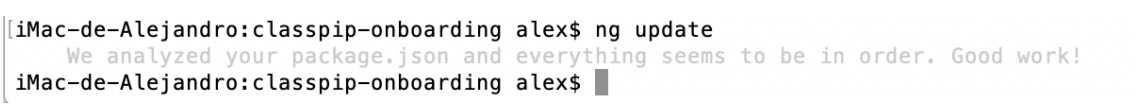
Introduint aquesta comanda, s'obté un llistat de les actualitzacions disponibles, si es que hi ha, de la següent manera:



El pas següent es introduir la comanda *ng update* que ens ofereix el mateix llistat d'actualitzacions una a una fins a haver-les introduïdes. El resultat es el que es veu a la següent imatge:

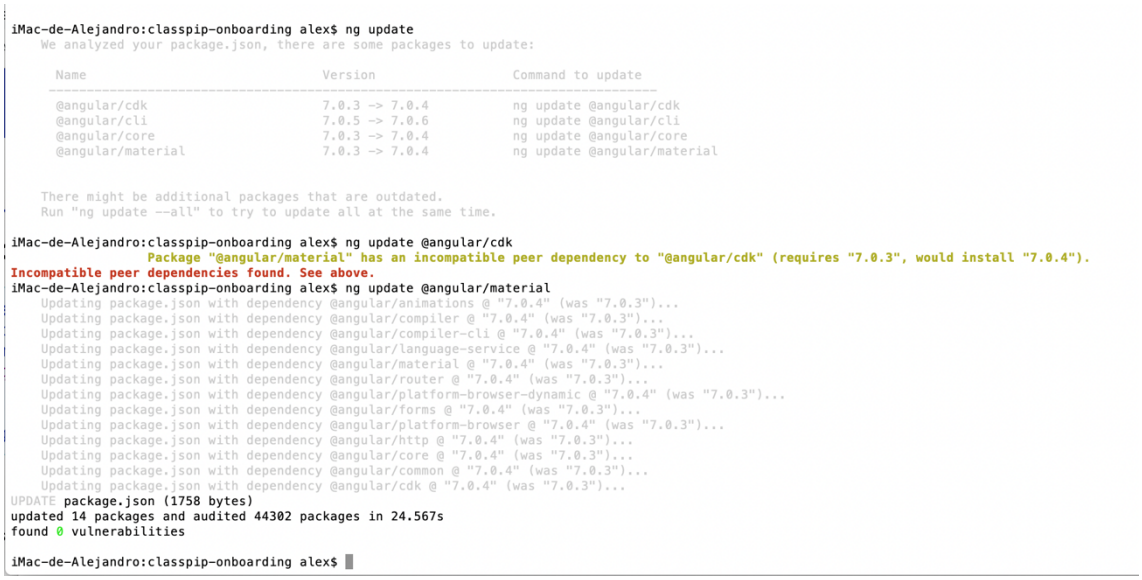


Un cop realitzades totes les actualitzacions, o si s'executa la comanda ng update sense haver actualitzacions disponibles, el resultat es el següent:



Important: en ocasions el resultat de l'actualització es un missatge d'error que avisa de que s'han trobat dependències incompatibles.

En la següent imatge es pot apreciar com a l'intentar actualitzar *@angular/cdk* de la versió 7.0.3 a la versió 7.0.4, apareix un missatge d'error informant de que *@angular/material* en la seva versió 7.0.3 requereix la versió 7.0.3 de @angular/cli. La solució a aquest problema es tan senzilla como realitzar la comanda d'actualització primer a *@angular/material*, y després a la resta de dependències, si es que no s'han actualitzat ja.



6.2.8 Compilar l'aplicació d'Angular

Molt important: després d'actualitzar o de fer un canvi a la web, i abans de realitzar el corresponent commit, s'ha de realitzar un *ng build* per a que els canvis generats també pugin a la web de producció.

```
ng build
```

6.3 Catàleg d'errors

6.3.1 Errors trobats a macOS

Error #1: Failed to install 'cordova-plugin-inappbrowser'

Error:

En la versió mobile al utilitzar la comanda *'cordova prepare'* s'obté l'error:

Failed to install 'cordova-plugin-inappbrowser':CordovaError: Failed to find 'ANDROID_HOME' environment variable. Try setting setting it manually.

Solució:

La forma de solucionar-ho es anar a la pàgina d'android studio i instal·lar l'Android SDK corresponent al sistema operatiu, en aquest cas la versió mac. Si es realitza la instal·lació de Android studio, ja ve l'Android sdk.

En cas de no voler utilitzar/instal·lar Android studio n'hi ha que descarregar només els arxius del sdk y copiar-los a la ruta /Users/alex/Library/Android/sdk. I després indicar-li al SO el PATH fins a aquests arxius amb les següents comandes.

```
export ANDROID_HOME=/Users/alex/Library/Android/sdk

export PATH=${PATH}:$ANDROID_HOME/tools:$ANDROID_HOME/platform-
tools
```

Error #2: repositories.cfg could not be loaded

Error:

Error: File \Users\User\.android\repositories.cfg could not be loaded.

Solució:

Crear arxiu repositories.cfg a la ruta \Users\User\.android\

Error #3: Build failed with an exception.

Error:

```
FAILURE: Build failed with an exception.

* What went wrong:
A problem occurred configuring root project 'android'.
> You have not accepted the license agreements of the following SDK components:
[Android SDK Platform 25].
Before building your project, you need to accept the license agreements and
complete the installation of the missing components using the Android Studio SDK
Manager.
```

Solució:

En Android studio anar al menú preferences i en la opció Android SDK a la pestanya sdk platforms i s'instal·la andoid 7.0 (Nougat) que es el que correspon a la plataforma 24. En la instal·lació s'accepten els acords de la llicencia.

Error #4: Errors de typescript quan s'executa ionic serve

Error:

Al executar la comanda *"ionic serve"* per iniciar el servidor del mòdul de mobile, aquest mostra tota una sèrie d'errors en typescript, obrint una finestra del navegador que mostra tots els errors de typescript.

Solució:

Al Package.json s'ha de pujar la versió de "typescript": "2.0.9" a la versió "typescript": "2.1.4". Després s'executa "npm install" un cop més. Ara al executar el servidor d'ionic no es mostra cap error i la pàgina de l'aplicació mobile s'obre sense cap problema.

Error #5: Errors derivats de la instal·lació de Node.js

Error:

Node.js s'instal·la correctament i sense donar cap error, però a l'hora d'utilitzar el client de comandes d'Angular, es comencen a veure multitud d'errors al terminal, i les comandes no funcionen. També s'obtenen errors al intentar instal·lar mòduls en el projecte amb npm install.

Solució:

El problema es que al realitzar una instal·lació de Node.js des de l'instal·lador que hi ha a la pagina de Node.js, aquest s'instal·la sense els permisos de lectura/escriptura corresponents. Al no tenir els permisos que necessita, al intentar realitzar les accions que requereixen aquests permisos no es poden executar i el funcionament es inestable o directament dona errors.

La millor forma de solucionar-ho es utilitzar l'instal·lador nvm (node version manager), s'instal·la mitjançant brew, amb la comanda "brew install nvm".

Després amb posar "nvm install 8.6" ja s'instal·la la versió de node 8.6 i es configuren tots els permisos necessaris correctament.

```
nvm install <versión>  // instala la versión de node indicada

nvm uninstall <versión>  // desinstala una versión de node

nvm list  // lista las versiones de node instaladas

nvm use <versión>  // cambia la versión de node a la indicada
```

6.3.2 Errors que es poden trobar a qualsevol sistema operatiu

Error #1: error intern del servidor

Figura 7.11 Missatge d'error intern en el servidor

Si en el moment d'executar l'aplicació i interactuar surt el missatge superior, es pot deure a que el programa estigui apuntant al servidor de producció.

Si es vol desenvolupar, s'ha de comentar i descomentar les següents línies del fitxer: **/classpip-mobile/src/app/app.config.ts**

```
// public static get SERVER_URL(): string { return
'https://api.classpip.com'; } // PRO

public static get SERVER_URL(): string { return
'http://localhost:3000'; } // DEV
```

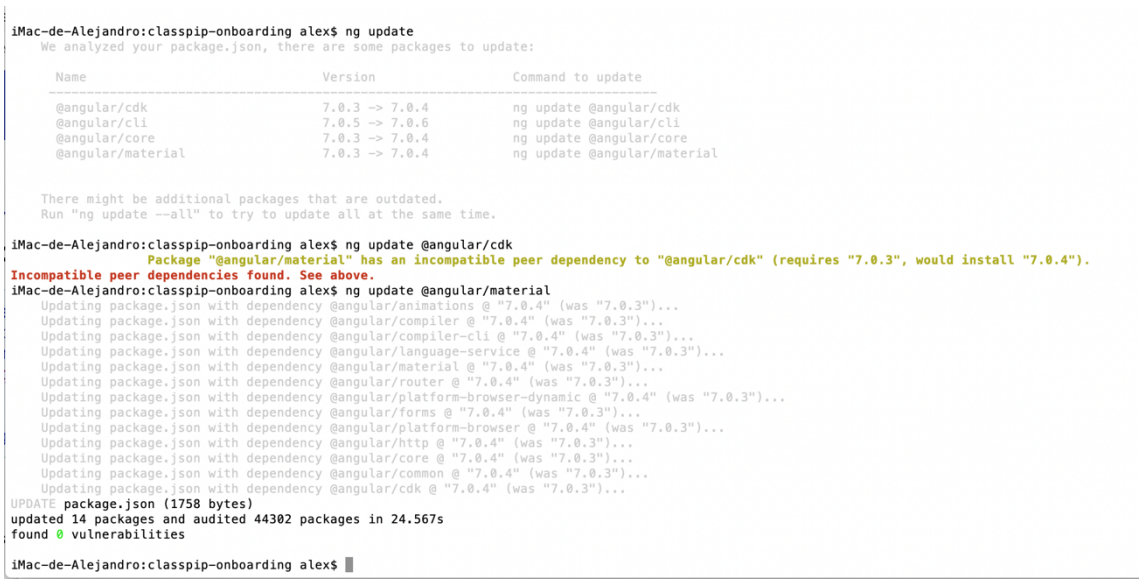
Un altre possible causa d'aquest error es que l'aplicació de serveis estigui tancada, en aquest cas només s'haurà d'executar l'aplicació de serveis per a solucionar el problema.

I en el cas de que l'aplicació de serveis estigui en marxa, i s'estigui utilitzant una base de dades en comptes de la memòria. S'haurà de comprovar que el servidor de mysql no hagi caigut. En cas d'haver caigut, basta amb reiniciar-lo per a solucionar el problema.

Error #2: error de dependències incompatibles al utilitzar ng update

En ocasions el resultat de l'actualització es un missatge d'error que avisa que s'han trobat dependències incompatibles.

En la següent imatge es pot apreciar com al intentar actualitzar @*angular/cdk* de la versió 7.0.3 a la versió 7.0.4, apareix un missatge d'error informant de que @*angular/material* en la seva versió 7.0.3 requereix la versió 7.0.3 de @angular/cli. La solució a aquest problema es tan senzilla com realitzar la comanda d'actualització primer a @*angular/material*, i després a la resta de dependències, si es que no s'han actualitzat també amb la primera comanda.



6.4 Actualització del dashboard a Angular 6

Al repositori del dashboard, n'hi ha una branca anomenada angular-update. En aquesta branca s'ha realitzat una actualització de tots els components a les ultimes versions del dia que es va realitzar aquesta tasca. Tot i que l'actualització va acabar funcionant correctament, aquesta no es va aplicar al projecte per a no perjudicar el funcionament de les aportacions que altres estudiants feien en paral·lel amb aquest projecte.

Per a realitzar aquesta actualització es van realitzar les següents peses. El primer es obrir un terminal i situar-se a la carpeta on esta el projecte de dashboard. Després s'han d'anar introduint les següents comandes al terminal en aquest mateix ordre:

```
npm install -g @angular/cli
npm install @angular/cli@latest
ng update @angular/cli
ng update
ng update @angular/core
ng update rxjs

rm -rf node_modules
npm i -g npm-check-updates
ncu -u
```

Ara, s'ha d'editar el package.json y canviar la versió de typeScrypt per "typescript": "2.9.2", i llavors es tornen a instal·lar els mòduls de node que s'acaben d'esborrar en una de les passes prèvies amb la següent comanda:

```
npm install
npm i rxjs-compat
```

En aquest punt es tenen diversos errors de compilació quan es tracta de posar en marxa el dashboard. Per solucionar-los s'han de canviar a tot el codi tots els:

```
import { Observable } from 'rxjs/Observable';
```

per aquest altre codi:

```
import { Observable } from 'rxjs/Rx';
import 'rxjs/add/operator/mergeMap';
```

Ara el que s'obté es un altre error:

```
ERROR in ./node_modules/ngx-lorem-ipsum/lib/ngx-lorem-
ipsum.component.ts
Module build failed: Error:
/Users/alex/Documents/proyectos/classpip/classpip-
dashboard/node_modules/ngx-lorem-ipsum/lib/ngx-lorem-
ipsum.component.ts is missing from the TypeScript compilation.
```

```
Please make sure it is in your tsconfig via the 'files' or
'include' property.
```

Per a poder solucionar-ho, s'ha d'anar a l'arxiu tsconfig.json, i en aquest arxiu, just a sota d'on posa *"compileOnSave": false,* s'ha d'escriure el següent:

```
"include": [
    "src/**/*.ts",
    "node_modules/ngx-lorem-ipsum/**/*.ts"
    ],
```

Ara tot sembla funcionar correctament, però tots els llocs on hi ha selects amb opcions per a triar, el codi ha deixat de respondre. En les darreres versions d'Angular ha deixat de funcionar el (change), ja que aquest event s'ha substituït per (selectionChange). Per aquesta raó a qualsevol lloc on hi hagi un selector com el següent:

```
<mat-select (change)="doSomething($event)">
```

S'ha de substituir per un com aquest:

```
<mat-select (selectionChange)="doSomething($event)">
```

Abans de realitzar l'actualització de versions del dashboard el llistat de vulnerabilitats es el següent:

```
[!] 54 vulnerabilities found [5914 packages audited]
    Severity: 28 low | 11 moderate | 15 high
    Run `npm audit` for more detail
```

Un cop realitzat el procés d'actualització, es pot comprovar com les vulnerabilitats del dashboard es redueixen considerablement.

```
added 1655 packages from 1432 contributors and audited 25711
packages in 47.156s
found 10 vulnerabilities (2 low, 5 moderate, 4 high)
run `npm audit fix` to fix them, or `npm audit` for details
```